[redacted]

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/ENY/95M-1

MULTIPOINT QUADRATIC APPROXIMATION FOR NUMERICAL OPTIMIZATION

THESIS

Michael A. Blaylock, Captain, USAF

AFIT/GA/ENY/95M-1

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 21,1995	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE Multipoint Quadratic Approximation for Numerical Optimization			5. FUNDING NUMBERS
6. AUTHOR(S) BLAYLOCK, Michael A., Capt, USAF			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT/ENY Maj Robert A. Canfield WPAFB OH 45433			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GA/ENY/95M-1
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Vipperla B. Venkayya WL/FIBA, Bldg 45 WPAFB OH 45431			10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A quadratic approximation for nonlinear functions is developed in order to realize computational savings in solving numerical optimization problems. Function and gradient information accumulated from multiple design points during the iteration history is used in estimating the Hessian matrix. The approximate Hessian matrix is the available for a second order Taylor series approximation to the functions of interest. Several truss and frame models will be used to demonstrate the effectiveness of the new Multipoint QuadraticApproximation (MQA) in solving structural optimization problems.

14. SUBJECT TERMS			15. NUMBER OF PAGES 110
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

AFIT/GA/ENY/95M

MULTIPOINT QUADRATIC APPROXIMATION FOR NUMERICAL OPTIMIZATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Michael A. Blaylock, B.C.E

Captain, USAF

March 1995

The views expressed in this thesis are those of the author and do not reflect the official policy of the Department of Defense or U.S. Government.

Abstract

A quadratic approximation for nonlinear functions is developed in order to realize computational savings in solving numerical optimization problems. Function and gradient information accumulated from multiple design points during the iteration history is used in estimating the Hessian matrix. The approximate Hessian matrix is the available for a second order Taylor series approximation to the functions of interest. Several truss and frame models will be used to demonstrate the effectiveness of the new Multipoint Quadratic Approximation (MQA) in solving structural optimization problems.

Acknowledgments

I would like to thank my thesis advisor, Major Robert Canfield, for his support and technical guidance throughout this research effort. His caring attitude helped me get through the tough times. I also owe a debt of gratitude to Ray Kolonay for letting me use his structural problems and computer code. I could not have finished in time without it.

I am grateful to many of my peers for their support throughout this project. One that I would like to thank specifically is Captain Robert N. Costa. Without his advice and moral support, I would have had a much harder time getting motivated.

I owe the greatest thanks to my wife Andi, who was always there for me no matter how tough things were. She carried more than her share, so that I could work on my thesis. Her support made the long nights bearable. Thanks.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENTS iii

LIST OF FIGURES vi

LIST OF TABLES vii

1. INTRODUCTION 1-1

1.1 OBJECTIVES 1-2
1.2 PROBLEM STATEMENT 1-2
1.3 SCOPE 1-2

2. BACKGROUND 2-1

2.1 OPTIMIZATION CONCEPTS 2-1
2.2 ITERATIVE PROCEDURE 2-3
2.3 ASSUMPTIONS 2-5

3. LITERATURE SURVEY 3-1

3.1 EARLY WORK 3-1
3.2 RECENT WORK 3-1
3.3 CONCLUSION 3-3

4. NEW THEORY 4-1

4.1 OVERVIEW 4-1
4.2 DIRECT VS. RECIPROCAL VARIABLES 4-1
4.2.1 Direct Variable Approximations *4-2*
4.2.2 Reciprocal Variables *4-3*
4.2.3 Using The Approximations *4-4*
4.3 ESTIMATING THE HESSIAN 4-5
4.3.1 Using All Previous Information *4-5*
4.3.2 Using Two Most Recent Points *4-14*
4.4 SPHERICAL MOVE LIMIT 4-15

5. NUMERICAL RESULTS 5-1

5.1 OVERVIEW 5-1
5.2 COMPARISON OF METHODS 5-3
5.2.1 Five Section Beam *5-3*
5.2.2 Four Member Frame *5-9*
5.2.3 Ten Bar Truss *5-16*
5.2.4 12 Member Frame *5-22*
5.2.5 ACOSS II (6 Design Variables) *5-29*
5.2.6 ACOSS II (31 Design Variables) *5-35*
5.2.7 40 Member Frame *5-42*

6. CONCLUSION 6-1

6.1 OVERVIEW 6-1
6.2 RECIPROCAL VS. DIRECT 6-1
6.3 QUADRATIC VS. LINEAR 6-2
6.4 SEQUENTIAL TWO POINT QUADRATIC VS. MULTIPOINT QUADRATIC 6-2
6.5 SPHERICAL VS. BOX MOVE LIMITS 6-3
6.6 RECOMMENDATIONS 6-3

6.7 THE BOTTOM LINE 6-3

APPENDIX A: MULTIPOINT QUADRATIC APPROXIMATION ROUTINE A-1

APPENDIX B: ROUTINE FOR CALCULATING THE APPROXIMATE HESSIAN .. B-1

APPENDIX C: ROUTINE FOR CALCULATING THE APPROX. FUNCTIONS C-1

APPENDIX D: ROUTINE FOR CALCULATING THE APPROX. GRADIENTS D-1

BIBLIOGRAPHY BIB-1

VITA V-1

List of Figures

FIGURE 2-1: OPTIMIZATION USING APPROXIMATIONS........ 2-3
FIGURE 5-1: 5-SECTION BEAM........ 5-3
FIGURE 5-2: DIRECT METHOD OPTIMIZATION........ 5-5
FIGURE 5-3: RECIPROCAL METHOD OPTIMIZATION........ 5-7
FIGURE 5-4: 4-MEMBER FRAME........ 5-9
FIGURE 5-5: DIRECT VARIABLE OPTIMIZATION........ 5-12
FIGURE 5-6: 4-MEMBER FRAME RECIPROCAL METHOD OPTIMIZATION........ 5-14
FIGURE 5-7: TEN BAR TRUSS........ 5-16
FIGURE 5-8: TEN BAR TRUSS DIRECT OPTIMIZATION........ 5-18
FIGURE 5-9: TEN BAR TRUSS OPTIMIZATION........ 5-20
FIGURE 5-10: 12 MEMBER FRAME........ 5-22
FIGURE 5-11: 12 MEMBER FRAME DIRECT OPTIMIZATION........ 5-25
FIGURE 5-12: 12 MEMBER FRAME RECIPROCAL METHOD OPTIMIZATION........ 5-27
FIGURE 5-13: ACOSS II........ 5-29
FIGURE 5-14: ACOSS II (6) DIRECT METHOD OPTIMIZATION........ 5-32
FIGURE 5-15: ACOSS II (6) RECIPROCAL METHOD OPTIMIZATION........ 5-34
FIGURE 5-16: ACOSS II (31) DIRECT METHOD OPTIMIZATION........ 5-38
FIGURE 5-17: ACOSS II (31) RECIPROCAL OPTIMIZATION........ 5-40
FIGURE 5-18: 40-MEMBER FRAME........ 5-42
FIGURE 5-19: 40-MEMBER FRAME DIRECT OPTIMIZATION........ 5-47
FIGURE 5-20: 40-MEMBER FRAME RECIPROCAL OPTIMIZATION........ 5-49

List of Tables

TABLE 5-1 5-SECTION BEAM OPTIMIZATION OVERVIEW 5-4
TABLE 5-2: DIRECT METHOD CONVERGENCE........ 5-6
TABLE 5-3: RECIPROCAL METHOD CONVERGENCE........ 5-8
TABLE 5-4: 4-MEMBER LOAD CASES 5-10
TABLE 5-5: 4-MEMBER FRAME MATERIAL PROPERTIES........ 5-10
TABLE 5-6: 4-MEMBER ELEMENT PROPERTIES........ 5-10
TABLE 5-7: 4-MEMBER OPTIMIZATION OVERVIEW 5-11
TABLE 5-8: DIRECT METHOD CONVERGENCE........ 5-13
TABLE 5-9: 4-MEMBER FRAME CONVERGENCE........ 5-15
TABLE 5-10: TEN BAR TRUSS OPTIMIZATION OVERVIEW 5-17
TABLE 5-11: TEN BAR TRUSS CONVERGENCE........ 5-19
TABLE 5-12: TEN BAR TRUSS CONVERGENCE........ 5-21
TABLE 5-13: 12 MEMBER FRAME LOAD CASES........ 5-23
TABLE 5-14: 12 MEMBER FRAME MATERIAL PROPERTIES........ 5-23
TABLE 5-15: ELEMENT PROPERTIES........ 5-24
TABLE 5-16: 12 MEMBER FRAME OPTIMIZATION OVERVIEW 5-24
TABLE 5-17: 12 MEMBER FRAME CONVERGENCE 5-26
TABLE 5-18: 12 MEMBER FRAME CONVERGENCE 5-28
TABLE 5-19: ACOSS II (6) OPTIMIZATION OVERVIEW........ 5-30
TABLE 5-20: ACOSS II (6) LINKING........ 5-31
TABLE 5-21: ACOSS II (6) CONVERGENCE........ 5-33
TABLE 5-22: ACOSS II (6) CONVERGENCE........ 5-35
TABLE 5-23: ACOSS II (31) OPTIMIZATION OVERVIEW 5-36
TABLE 5-24: ACOSS II (31) LINKING........ 5-37
TABLE 5-25 ACOSS II (31) CONVERGENCE 5-39
TABLE 5-26: ACOSS II (31) CONVERGENCE 5-41
TABLE 5-27: 40-MEMBER FRAME LOAD CASES........ 5-43
TABLE 5-28: 40-MEMBER FRAME MATERIAL PROPERTIES........ 5-45
TABLE 5-29: 40-MEMBER FRAME ELEMENT PROPERTIES 5-45
TABLE 5-30: 40-MEMBER FRAME OPTIMIZATION OVERVIEW 5-46
TABLE 5-31: 40-MEMBER FRAME CONVERGENCE 5-48
TABLE 5-32: 40-MEMBER FRAME CONVERGENCE 5-50
TABLE 6-1: OPTIMIZATION SUMMARY........ 6-1

1. Introduction

The standard nonlinear optimization problem is to minimize a function $f(X)$ that depends on a design variable vector X. This function is minimized in a feasible region Ω defined by inequality and equality constraints $g(X)$ and $h(X)$, respectively. Recently, a more efficient approach has been developed using intermediate design variables Y and response quantities $q(Y)$ that may be better approximated than the actual functions, but still can be easily related back to the original functions and design variables. (Schmit,1977:19-24), (Barthelemy,1991), and (Starnes,1979:564-570) The problem now is to minimize the new functions related to $f(X)$, $g(X)$, $h(X)$, and Ω based on the better approximated design variable and response functions. To carry out the exact calculation of the response quantities is typically quite expensive, so they are normally approximated using a linear Taylor series approximation in the intermediate design variable Y of the form: $\tilde{q}(Y) = q(Y_0) + \nabla q^T(Y_0)\Delta Y$. The values of the approximate functions are then calculated and used to converge by iteration to a suitable tolerance of the exact solution. Using this iterative approach one can drastically reduce the number of full analyses, provided the approximations are of high enough quality. This approach and many similar approaches all use the linear Taylor series approximation to the response function based on a single design point. New approximations are constructed each time an exact analysis is made. Information from previous analyses is typically lost. It has been shown that higher accuracy can be achieved with a quadratic Taylor series approximation (Fleury,1986:409-428); however, the computational expense of calculating second derivatives often outweighs the benefit gained from faster convergence of the more accurate estimates. If a computationally less expensive method could be developed for obtaining approximate second derivatives, then faster convergence should be obtained.

1.1 Objectives

The objective of this thesis is to test a method of numerical optimization that uses a quadratic approximation for the objective function and constraints. The quadratic approximation will be developed using a Hessian matrix that is obtained from the exact analyses at each of the previous points. As more points are obtained near the solution, the approximation for the Hessian will improve. Also, the Hessian will be calculated weighing the information from closer points more heavily. It is expected that using this quadratic approximation will improve the accuracy of the approximations that are sent to the optimizer, resulting in faster overall convergence. A side benefit that is expected from the curvature obtained from using a quadratic approximation is that the dependence of optimization problems on move limits will be reduced. A second quadratic method Sequential Quadratic Approximation (SQA) will also be tested.

1.2 Problem Statement

The overall problem is to find a way to mimic a second order Taylor series approximation without actually having the expense of calculating second derivatives. The method used estimates of the Hessian matrix by using information that is already available from previous analyses.

1.3 Scope

The scope of this thesis is to develop this multipoint quadratic approximation, then compare it to other methods to discover if it can solve actual problems with fewer iterations. A wide range of structural problems have been chosen for comparison, so that some idea of the various method's strengths and weaknesses can be obtained. These structural problems are of various sizes ranging from a five section beam with one constraint to a 40 member frame with 408 constraints. These problems also contain a variety of constraints including stress constraints, displacement constraints and frequency constraints.

2. Background

Typically in engineering we want to produce the best design possible based on the constraints given to us. Optimization concepts have provided a way to systematically produce the best design based on known parameters. A typical example is to design the lightest structure that will carry a prescribed load. The weight of the structure is the function that we are trying to minimize and the allowable stresses and displacements are the constraints. We can make the structure as light as possible but we must do so in a manner that does not cause it to collapse. In the past, this was done based on experience and trial and error. Numerical optimization techniques have now given engineers systematic approaches to solving these problems.

2.1 Optimization Concepts

The best way to explain the optimization process is through a simple example. The first step in the optimization problem is determining what is to be minimized (or maximized); for example, minimizing the weight of a structure. In order for the problem to be optimized, an equation or function must be developed for the weight. This function is the objective function, which will be called $f(X)$. The objective function is a function of parameters called design variables X. These design variables typically are the cross sectional areas of the members of a structure provided that the lengths of the members are fixed. The next step is to develop constraint functions also based on the same design variables. An example of a constraint would be the allowable stress in a member. A function must be developed relating the cross sectional area (the design variables) to the allowable stresses. These are our constraint functions. There are also constraints called equality constraints that relate directly to the design variables. Equality constraints are typically explicit functions of X that relate one of the design variables to another. An example of the would be a constraint where the cross sectional the depth of a beam has to be twice its width. Now, we have an objective function *f(X)*, constraint functions *g(X)*, and equality constants *h(X)* with *X* being the vector of design variables.

The objective is to minimize *f(X)* while not violating any of the constraints in *g(X)* and *h(X)*. For the rest of this thesis, equality constraints *h(X)* are not used in the numerical examples, although this is not an inherent restriction. The next question is, "how do we know when a minimum has been achieved".

The three necessary and sufficient conditions for a minimum are called the Kuhn-Tucker conditions. (Vanderplats,1984:17) These are:

1. X is feasible
2. $\lambda_j g(X)_j = 0 \quad j=1,m \qquad \lambda_j \geq 0$
3. $\nabla f(X) + \sum_{j=1}^{m} \nabla g_j(X) = 0 \quad \lambda_j \geq 0$

If a constraint is not active, i.e., not equal zero, then its λ is zero. Using these conditions, an optimizer solves a problem given a function and it constraints. However, it must be pointed out that this method only finds a local minimum of a function. If more than one local minimum exists, the optimizer could find any of one of them. There is no guarantee that the one it finds will be the global minimum. For this thesis it is assumed that the optimizer can do its job effectively. The objective in this thesis is to determine what is the best form to express the objective function and constraint function so that the optimizer can solve the overall problem quickly and efficiently.

2.2 Iterative Procedure

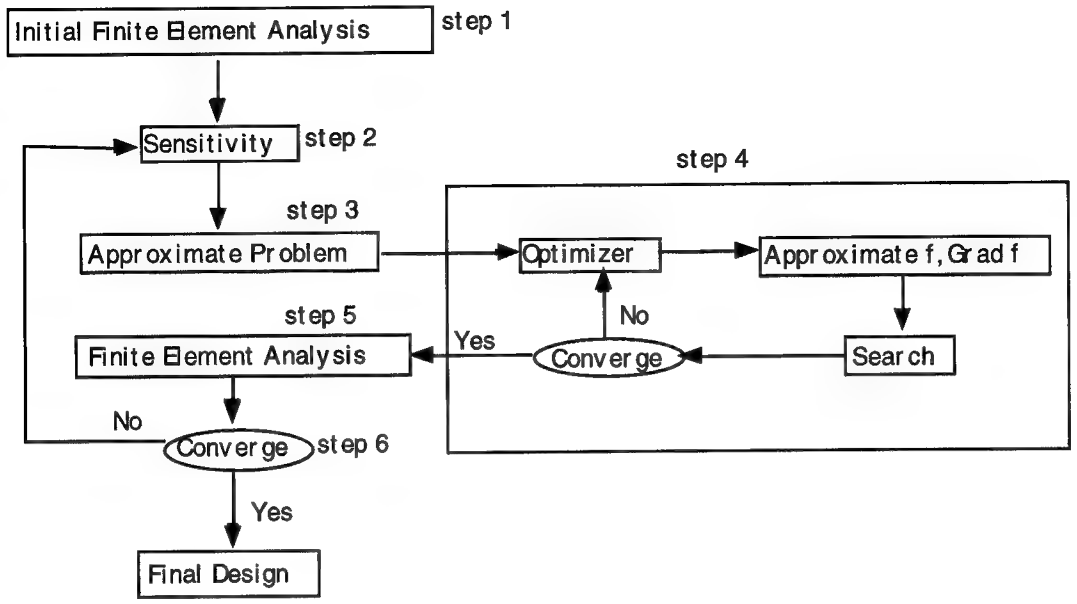


Figure 2-1: Optimization Using Approximations

For structural optimization, the overall procedure for the "approximation concepts" approach is shown in Figure 2-1. This is the approach used by many commercial structural analysis programs that also include optimization, such as MSC/NASTRAN, IDEAS, and ASTROS. The first step is to perform the initial analysis based on some starting point. It is generally best to have this starting point be a feasible solution to the problem but it is not necessary. Starting at a highly infeasible point far from the optimum can make the optimization take longer. The initial analysis must give us the value of the objective function and the values of all the constraints, *f(X)* and *g(X)*, respectively. The second step is to do the sensitivity analysis to obtain the derivatives of the objective function and constraints. The third step is the approximate problem generator, the focus of this thesis. This step could be skipped and the exact problem could be given to the optimizer so that it can solve the problem to find the optimum. The disadvantage with this approach is that this typically requires many analyses. If each analysis takes a lot of computer time then it will take a long time to find the optimum. A better way is to approximate the exact problem with an explicit equation that gives us nearly the same values of the objective function, constraints and the gradients for each. These approximations will only be good near the point, therefore move limits are placed on the design variables. Move limits are restrictions on how much the optimizer can vary the design variables to find the optimum. Move limits are added as another constraint. For example, if we give the optimizer a starting point of two design variables of (1,2) and impose a move limit of 20%, this means that the optimizer is restricted to use values of the design variable in the range of (.8-1.2,1.6-2.4). Move limits of this type are called box move limits. Another type of move limit is a circular or spherical move limit for which changes in one design variable limit changes in another. The way this is imposed is by taking the dot product of the normalized design variable vector with itself and setting this value equal to some parameter, given by

$$\Delta X^T S \Delta X \le r$$

where ΔX is the change in the design variable, S is a scaling factor matrix to normalize the variables and r is the move limit, such as 0.2. S is a square diagonal matrix with one divided by the square root of the original x value on its diagonal for each contraint. For example, if X=(1,2) then

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sqrt{1}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \le 0.2$$

From this one can see the size of the change in one variable limits the change in the size of another. Instead of being a box this is the equation of a circle. In multidimensional space the box becomes a cube or hypercube and the circle becomes a sphere or hypersphere. One of the objectives of this thesis will be to see if spherical move limits have an advantage over box type move limits.

Once the move limits are established, the approximate problem is then given to the optimizer. The optimizer then proceeds to find the approximate optimum solution within the bounds of the move limits and constraints. Once it finds this intermediate design point, the new X vector is used in the next step where a detailed analysis is performed at the new point. In the next step the actual objective function value and the actual constraints are compared to the approximate optimum received from the optimizer. If these are within specified tolerance of each other, then the optimization process is complete. If not, then a new sensitivity is computed based on the new point that should be closer to the optimum than the previous point. A new approximate problem is then created and this is given to the optimizer to solve. Once solved, convergence is again checked. With each iteration, the routine should get closer to the true optimum. One problem often encountered is that the solution gets close to the optimum but has trouble converging. This is because the accuracy of the approximation and the convergence criteria may not be of the same order. To prevent this problem, the approximation must be made more accurate. This can be done by systematically reducing the move limit, since the closer we are to the point of the last actual calculation, the more accurate the approximation. The key factor with move limits is to have them large enough when optimization begins so that the approximation can move closer to the optimum at a reasonable rate, and having them small enough at the end so the approximation is accurate enough to converge. However, as the accuracy of the approximation increases the dependence on having the proper (small enough) move limits should decrease.

2.3 Assumptions

This thesis focuses strictly on step three in Figure 2-1, the approximate problem generator. It is assumed that the programs that calculate the analysis and sensitivities are correct. Also, it is assumed that the optimizer does its job efficiently. The optimizer used was "constr" in MATLAB (Grace,1993:9-11), and for the most part it worked well. However, sometimes it had trouble solving the approximate problem. This is outside the scope of this thesis because the approximate problems developed here are generic and could easily be adapted to work with any such optimizer.

3. Literature Survey

Figure 2-1 shows the typical optimization process. The focus of this thesis is on step 3, which is generating the approximate problem for the optimizing algorithm to solve. The basic concept is that the more accurate the approximate problem, the faster the problem will converge to the optimum.

3.1 Early Work

In the 1950's the first implementation of mathematical programming was used by Heyman and Livesly to optimize a single bay structure subject to a single load case (Kolonay,1987:6). It quickly became obvious that optimizing a structure involved numerous iterations and would involve numerous solutions to reanalyze the structure and to calculate the constraints. In the 1960's, Moses developed the technique of using a Taylor series expansion to approximate the objective function and constraints. (Kolonay,1987:6) By doing this, he reduced the number of analyses required to solve the optimization problem. The typical optimization problem used a first order Taylor series approximation using the information from the last full analysis. A first order approximation was used because calculating second derivatives needed for the Hessian matrix for the second order approximation was very computationally expensive.

3.2 Recent Work

Recent work in the area of improving the approximation used by the optimizer in structural optimization has taken several directions. The first direction is with coming up with intermediate design variables such as reciprocal design variables that can form better approximate nonlinear, but explicit, functions than the original design variables. The second direction is using information from more than one analysis or design point. An application of the second direction is to use previous information to develop a quadratic approximation. Pertinent to both of these methods is the question of move limits. Move limits are how far we let the optimizer go in using the approximation. A Taylor series approximation is an approximation about or near a point; therefore, the farther away from the point the worse the approximation. Determining how far away from the point the approximation is adequate is a science and art all to itself. However, if one can increase the distance where the approximation is valid, this should also increase the rate of convergence.

It was discovered that by using Taylor series approximations in reciprocal variables, the accuracy of the approximations improved and faster convergence was achieved. (Schmit,1974:692-699). As Vanderplaats states, "using reciprocal design variables can make non-linear structural optimization problems linear, so that a Taylor series approximation is very accurate" (Vanderplaats,1984,260-263). This technique has reduced the number of analyses required in many structural problems and it worked well because the quantities being reciprocated (such as cross sectional areas) are positive and not near zero. In some optimization problems where a variable could be zero, the reciprocal approximation could become undefined.

Typically the information used to develop the approximation was only based on the current point and the information from earlier analyses was lost. Haftka and coworkers initially investigated multipoint approximations with mixed results (Haftka,1987:289-301). In particular, they concluded that two and three point approximations may be no better than the reciprocal approximations for structural problems. Fadel incorporated previous gradient information in a two point approximation that was used to select better intermediate design variables for a linear approximation (Fadel,1990,117-124). In contrast, Rasmussen incorporated only previous function values, while ignoring gradients from previous iterations (Rasmussen,1990:253-258). He demonstrated stable convergence for a cantilever box beam where other methods failed to converge. Recently, Wang and Grandhi demonstrated more rapid convergence for frame structures by employing multivariate splines based on all available exact analyses (Wang,1994:2090-2098). However, their method does not always reproduce known function values. In another paper with the help of Canfield they did reproduce function and gradient values (Wang,1994:269-280).

Miura and Schmit demonstrated the higher accuracy of quadratic Taylor series approximations for structural optimization with frequency constraints (Miura,1978:336-351). However, the computational savings due to faster convergence were balanced by the increased cost of calculating second derivatives. Snyman and Stander had success in using a two point successive approximation which uses a pseudo second order approximation based on these two points (Snyman,1994). The Hessian calculated for this case was diagonal with all terms on the diagonal being the same value. The intent was not to estimate the exact Hessian but to give the optimizer some curvature information at the new point. The benefits of this approach were reduced dependence on move limits and slightly faster convergence. Also, a large amount of Hessian information did not need to be stored. In 1993 Toropov, Filatov, and Polynkin developed a method of using multiple point approximation in a subregion to where finite element analyses are performed, then explicit approximations of the objective and constraint function are obtained and sent to the optimizer to be optimized. The subregions are selected based on the move limits. Their method uses weighting factors that give values closer to known points more weight (Toropov,1993:7-14).

3.3 Conclusion

All of the methods discussed above are a step towards a general method to develop a better approximation for the optimizer to use. An approach that could combine the best of these could possibly be more general and robust. If one could use reciprocal design variables and put the information obtained from all previous analyses to use, it would seem feasible that a better approximation could be found and used for faster convergence of the optimization problem.

4. New Theory

4.1 Overview

The focus of this thesis is to develop and test a new approximation method to be used by an optimization routine. The basic concept is to make use of the information obtained in each analysis to provide a more accurate approximation of a function. Typical optimization routines use a first order Taylor series approximation because it is too computationally expensive to use the actual function. Also, the approximation typically is based only on the latest point. The information from previous analyses is no longer of any use. The approach used here is to use the information obtained in all previous analyses to create a quadratic approximation of the function. This is done without calculating second derivatives. A side development during this research was the creation of spherical move limits that could be used with any approximation routine.

To compare whether any benefit is obtained from this quadratic approach, it will be compared to other methods already being used. All of the optimization routines use one of four types of approximations. The first type is using direct variables and a linear approximation, the second is using direct variables and a quadratic approximation, the third is using reciprocal variables with a linear approximation and the fourth is reciprocal variables with a quadratic approximation.

4.2 Direct vs. Reciprocal Variables

Direct variables are usually physical quantities in the optimization problem, such as the cross sectional areas of a structural member. A common objective function in a structural problem is the weight of a structure which is linear in the cross sectional area. This is the case for all of the problems in this research. Since the objective functions are linear, a first order Taylor series approximation is exact. However, the constraints for structural problems are usually not linear functions. A typical constraint for a structural problem is a maximum allowable stress, which is a function of force divided by area. Therefore, a linear approximation of the constraint is probably not very accurate. If, instead of using the cross sectional area as the design variable, one uses an intermediate design variable that is equal to one divided by the cross sectional area, then a better approximation can be achieved. The stress then becomes a linear function of this new design variable. Such variables are called reciprocal variables (Vanderplats,1984:260-263). Their use can greatly improve the accuracy of a Taylor series approximation for many structural problems. However, if the values of the design variables get very small or pass through zero, using reciprocal variables can result in functions becoming undefined. In these cases, reciprocal variables will not be very useful.

4.2.1 Direct Variable Approximations

The first order Taylor series approximation of the multivariate function $f(\mathrm{x})$ near some point x_0 is

$$\tilde{f}_L(\mathrm{x}) = f(\mathrm{x}_0) + [\nabla \mathrm{f}(\mathrm{x}_0)]^T \Delta\mathrm{x} \tag{4-1}$$

with x being a vector of the design variables. Two of the methods used in this thesis use this approximation. They are SPA-DS, which stands for single point approximation using direct variables with spherical move limits and SPA-DB which is a single point approximation using direct variables with box move limits. SPA-DB is the same as what is usually called sequential linear programming (SLP). The quadratic Taylor series approximation contains all the information of the linear plus a second order term. It is of the form:

$$\tilde{f}_Q(\mathrm{x}) = f(\mathrm{x}_0) + [\nabla \mathrm{f}(\mathrm{x}_0)]^T \Delta\mathrm{x} + \frac{1}{2}\Delta\mathrm{x}^T \mathrm{H} \Delta\mathrm{x} \tag{4-2}$$

where H is the Hessian matrix. The approach tested in this research is to develop a reasonable approximation for the true Hessian based on previous points. One method tested uses this approximation; it is called MQA-D, which stands for multipoint quadratic approximations using direct variables.

4.2.2 Reciprocal Variables

To develop the first order single point approximation in reciprocal variables we begin with the first order Taylor series approximation in the intermediate variable y

$$\tilde{f}_L(\mathrm{y}) = f(\mathrm{y}_0) + [\nabla \mathrm{f}(\mathrm{y}_0)]^T \Delta \mathrm{y} \tag{4-3}$$

which is the same as eq. (4-1) except using y instead of x.

For reciprocal variables $y = y(x)$, where $y_i = \dfrac{1}{x_i}$, we define the single point approximation with respect to x_i as

$$\tilde{f}_R(x) = f_L(y(x)) \tag{4-4}$$

where

$$f_y(y_0) = f(x_0) \tag{4-5}$$

and

$$\nabla_y f(y_0)\Delta y = \sum_{i=1}^{n} \frac{\partial f}{\partial y_{0i}}(y_i - y_{0i}) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_{0i}} \frac{\partial x_{0i}}{\partial y_{0i}} \left(\frac{1}{x_i} - \frac{1}{x_{0i}} \right)$$

for which n is the number of design variables (length of the x vector).

$$\frac{\partial x_{0i}}{\partial y_{0i}} = -\frac{1}{y_{0i}^2} = -x_{0i}^2$$

Therefore the final equations for the reciprocal linear approximation for x is

$$\tilde{f}_R(x) = f(x_0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_{0i}} \left(-x_{0i}^2\right)\left(\frac{1}{x_i} - \frac{1}{x_{0i}} \right). \tag{4-6}$$

This approximation is used by SPA-RS and SPA-RB which are single point approximations using reciprocal variables and spherical and box move limits, respectively.

The quadratic approximation for reciprocal variables starts with the linear approximation and adds another term:

$$\tilde{f}_{QR}(x) = f(x_0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_{0i}}\left(-x_{0i}^2\right)\left(\frac{1}{x_i} - \frac{1}{x_{0i}}\right) + \frac{1}{2}\Delta y^T H_y \Delta y. \qquad (4\text{-}7)$$

The Hessian approximations can estimate the Hessian with respect to y, therefore no chain rule is required when converting the second order term to use variables in x. The final equation is

$$\tilde{f}_{QR}(x) = f(x_0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_{0i}}\left(-x_{0i}^2\right)\left(\frac{1}{x_i} - \frac{1}{x_{0i}}\right) + \frac{1}{2}\left(\frac{1}{x_i} - \frac{1}{x_{0i}}\right) H_{yij}\left(\frac{1}{x_i} - \frac{1}{x_{0i}}\right). \qquad (4\text{-}8)$$

Eq. (4-8) is the equation used by MQA-R, multipoint quadratic approximation using reciprocal variables and by SQA, the sequential quadratic approximation.

4.2.3 Using The Approximations

Now that these approximations have been developed, the next step is to use them to solve an optimization problem. The approximation for the objective function *f(X)* is $\tilde{f}(X)$ and could be any of the four previously mentioned, although typically the approximation of the objective function is linear in direct variables. The approximation for the constraint function *g(X)* is $\tilde{g}(X)$. There will be an approximation for each constraint. The problem now becomes one of finding

$$\tilde{f}(x^*) = \min \tilde{f}(x),\ x \in \tilde{\Omega} \qquad (4\text{-}9)$$

$$\tilde{\Omega} = \{x | \tilde{g}(x) \le 0\} \qquad (4\text{-}10)$$

where Ω is a feasible region where none of the constraints are violated. These equations are solved iteratively until the approximations converge to the exact solutions to within a certain tolerance.

4.3 Estimating the Hessian

4.3.1 Using All Previous Information

Canfield developed the approach which is to be tested (Canfield,1994). The following parallels his derivation. We begin with a quadratic Taylor series approximation of a function of x.

$$\tilde{f}_Q(\mathrm{x},\mathrm{H}) = f(\mathrm{x}_k) + [\nabla \mathrm{f}(\mathrm{x}_k)]^T \Delta\mathrm{x} + \frac{1}{2}\Delta\mathrm{x}^T \mathrm{H} \Delta\mathrm{x} \qquad (4\text{-}11)$$

Where x_k is the design vector at the *k*th iteration, $\Delta\mathrm{x} = \mathrm{x}_k - \mathrm{x}_{k-1}$, is the change in the design vector, $\nabla \mathrm{f}(\mathrm{x}_k)$ is the function gradient, and H is an approximation to its Hessian. If H were composed of analytic second derivatives, eq.(4-11) would represent a second order Taylor series approximation. Instead of calculating the actual Hessian we will use the information obtained at previous points to calculate the Hessian required for the function to pass through these points and to match the gradients as closely as possible. After each iteration more information is available and a new Hessian is calculated using the additional information. Also, points that are closer to the current point are given more weight because closer information should give the best approximation. As we get closer to the optimum the approximate Hessian should become closer to the actual Hessian and should speed convergence. The first step is to develop a set of equations that can be solved to find the elements of the Hessian. We will begin with a first order Taylor series approximation as shown eq.(4-12).

$$\begin{gathered} f_L(\mathrm{x}_p) = f(\mathrm{x}_k) + [\nabla \mathrm{f}(\mathrm{x}_k)]^T (\mathrm{x}_p - \mathrm{x}_k) \\ p \in P = 1,2,\ldots,k-1 \end{gathered} \qquad (4\text{-}12)$$

Usually the first order approximation is not equal to the actual function value. However, if these two values are known, such as at previous points, then the elements of the Hessian can be considered unknown parameters which can be found by the known error. Eq.

(4-13) sets the quadratic approximation equal to the known function at the previous points. The only unknowns are the elements of the Hessian matrix. It is obvious that this problem is underdetermined until sufficient points are obtained.

$$\tilde{f}_Q(\mathrm{x}_p,\mathrm{H}) = f(\mathrm{x}_p) = f(\mathrm{x}_k) + [\nabla \mathrm{f}(\mathrm{x}_k)]^T(\mathrm{x}_p - \mathrm{x}_k) + \frac{1}{2}(\mathrm{x}_p - \mathrm{x}_k)^T \mathrm{H}(\mathrm{x}_p - \mathrm{x}_k) \qquad (4\text{-}13)$$
$$p \in P = 1,2,\ldots,k-1$$

Considering that the symmetric Hessian matrix has N=n(n+1)/2 unknown elements for n design variables, we need more equations to solve for these unknowns. Therefore, we would also like to reproduce the gradients from previous design points as closely as possible. The gradient of the quadratic approximation, eq. (4-14) evaluated at the previous design points is

$$\nabla_{\mathrm{x}} \tilde{f}_Q(\mathrm{x}_\mathrm{p},\mathrm{H}) = \nabla \mathrm{f}(\mathrm{x}_k) + \mathrm{H}(\mathrm{x}_p - \mathrm{x}_k) = \nabla \mathrm{f}(\mathrm{x}_\mathrm{p}), p \in P. \qquad (4\text{-}14)$$

Equations (4-13) and (4-14) constitute a set of linear equations in the N unknown elements of the Hessian matrix H. However, the equations are in general inconsistent. The equations are solved using the Moore-Penrose pseudo-inverse which provides a unique solution with the smallest norm of the terms in the Hessian.

The first step in solving (4-13) and (4-14) is to write them in standard form for solving a set of linear equations. Eq. (4-14) becomes

$$\mathrm{X}_p \mathrm{h}^R = \gamma_p, \quad p \in P \qquad (4\text{-}15)$$

where

$$\mathrm{X}_p = \begin{bmatrix} \Delta \mathrm{x}_p^T & 0 & \ldots & 0 \\ 0 & \Delta \mathrm{x}_p^T & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \Delta \mathrm{x}_p^T \end{bmatrix} \qquad (4\text{-}16)$$

X_p is an nxn^2 matrix and

$$\Delta x_p = x_p - x_k \tag{4-17}$$

is the change in x from each previous point to the current point where

$$\mathrm{h}^R \equiv \left\lfloor \lfloor H_{11}\ H_{12}\ \cdots H_{1n} \rfloor \lfloor H_{21}\ H_{22}\ \cdots H_{2n} \rfloor \cdots \lfloor H_{n1}\ H_{n2}\ \cdots H_{nn} \rfloor \right\rfloor^T \tag{4-18}$$

is a vector of the rows of H. The right hand side is the vector difference between the actual and approximate gradient for each previous design vector:

$$\gamma_p \equiv f(x_p) - \tilde{f}_L(x_p) \tag{4-19}$$

The next step is to put eq.(4-13) into standard form. It can be rewritten as

$$\frac{1}{2}\Delta \mathrm{x}_p^T \mathrm{X}_p \mathrm{h}^R = \delta_{p,}\ p \in P \tag{4-20}$$

where the right hand side is equal to the difference between the actual function value and its linear approximation.

$$\delta_p \equiv f(x_p) - \tilde{f}_L(x_p) \tag{4-21}$$

Once eqs.(4-13) and (4-14) have been put into this form, they are combined into a set of linear equations of the form

$$\mathrm{Xh}^R = \mathrm{b}. \tag{4-22}$$

where

$$\mathrm{X} \equiv \begin{bmatrix} \mathrm{X}_1 \\ \mathrm{X}_2 \\ \vdots \\ \mathrm{X}_p \\ \chi \end{bmatrix} \tag{4-23}$$

which is a matrix created from the left hand sides of eqs. (4-15) and (4-20) and for which

$$\chi \equiv \frac{1}{2}\begin{bmatrix} \Delta \mathrm{x}_1^T \mathrm{X}_1 \\ \Delta \mathrm{x}_2^T \mathrm{X}_2 \\ \vdots \\ \Delta \mathrm{x}_p^T \mathrm{X}_p \end{bmatrix} \tag{4-24}$$

The combined right hand side of eq. (4-22) is

$$\mathrm{b} \equiv \lfloor \gamma_1^T \;\; \gamma_2^T \;\cdots \gamma_p \;\; \delta_1 \;\; \delta_2 \;\ldots\; \delta_p \rfloor^T. \qquad (4\text{-}25)$$

We impose the condition that H be symmetric through the linear transformation:

$$\mathrm{h}^R = \mathrm{Th}^S \qquad (4\text{-}26)$$

where T is an n^2 x N linking matrix of zeros and ones, with one non-zero entry per row and where h^S is an N x 1 vector of the symmetric terms in H.

The next step is to calculate the weighting factors so that closer points are given more weight and we emphasize matching function values over gradients. D is the matrix containing all the weighting factors.

$$\mathrm{D} = \mathrm{diag}(\mathrm{W}_1 \;\; \mathrm{W}_2 \;\; \cdots \;\; \mathrm{W}_p \;\; \mathrm{W}) \qquad (4\text{-}27)$$

The W_p's are the weighting factors for each previous point and W is the weighting for functions over gradients. Thus

$$\mathrm{W}_p = \beta_p \mathrm{I}_{n \times n} \qquad (4\text{-}28)$$

in which I is a nxn identity matrix and β_p is a scalar weight inversely proportional to the square of the distance from the current point.

$$\beta_p = \left\| \Delta x_p \right\|_2^{-2} \qquad (4\text{-}29)$$

The weighting factor for function values is

$$\mathrm{W} = w\mathrm{diag}(\beta_1 \;\; \beta_2 \;\; \cdots \;\; \beta_p) \qquad (4\text{-}30)$$

where

$$w = \frac{\max\limits_{p \in P} \left\| \mathrm{W}_p \gamma_p \right\|_\infty}{\min\limits_{p \in P} \left| \delta_p \right|}. \qquad (4\text{-}31)$$

Once all the factors are calculated the final expression for the approximate Hessian becomes

$$h^R = T(DXT)^{\dagger} Db \tag{4-32}$$

The symbol † is used to represent the Moore-Penrose pseudo inverse.

To help illustrate the process of estimating the Hessian, we will estimate the Hessian for a simple function by using the equations and discussion above. Begin by picking a function, three previous points, and one current point.

The function is

$$f(X) = 1 + \frac{\sqrt{3}}{3x_1} - \frac{2}{\frac{x_1}{4} + x_2}$$

The current point x_k is (1,1). The three previous points are:

$$x_{p1} = \begin{pmatrix} .5 \\ .5 \end{pmatrix} \quad x_{p2} = \begin{pmatrix} .75 \\ .75 \end{pmatrix} \quad x_{p3} = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}$$

The function values at these points are:

$$\begin{aligned} f(x_k) &= -0.02265 \\ f(x_{p1}) &= -1.0453 \\ f(x_{p2}) &= -0.36353 \\ f(x_{p3}) &= 0.31823 \end{aligned}$$

The gradient of $f(X)$ is:

$$\nabla f(X) = \begin{bmatrix} \dfrac{-\sqrt{3}}{3x_1} + \dfrac{1}{2\left(\dfrac{x_1}{4} + x_2\right)^2} \\ \dfrac{2}{\left(\dfrac{x_1}{4} + x_2\right)} \end{bmatrix}$$

The gradient at each point is

$$\nabla f(x_k) = \begin{pmatrix} -0.257 \\ 1.28 \end{pmatrix}, \quad \nabla f(x_{p1}) = \begin{pmatrix} -1.03 \\ 5.12 \end{pmatrix}, \quad \nabla f(x_{p2}) = \begin{pmatrix} -0.458 \\ 2.28 \end{pmatrix}, \quad \nabla f(x_{p3}) = \begin{pmatrix} -0.114 \\ 0.569 \end{pmatrix}.$$

From the above function and gradients an approximation to the Hessian at point x_k will be made using the following equations, where P is the number of previous points, which is 3 for this case. Now, we will put eq. (4-14) in standard form

$$\mathrm{X}_p \mathrm{h}^R = \gamma_p, \quad p \in P \tag{4-33}$$

using eqs. (4-16), (4-11), (4-18) and (4-19).

From eq. (4-16)

$$\mathrm{X}_{p1} = \begin{bmatrix} -.5 & -.5 & 0 & 0 \\ 0 & 0 & -.5 & -.5 \end{bmatrix}, \mathrm{X}_{p2} = \begin{bmatrix} -.25 & -.25 & 0 & 0 \\ 0 & 0 & -.25 & -.25 \end{bmatrix}, \mathrm{X}_{p3} = \begin{bmatrix} .5 & .5 & 0 & 0 \\ 0 & 0 & .5 & .5 \end{bmatrix}$$

and eq.(4-19) we get these quantities:

$$\gamma_{p1} = \begin{pmatrix} -0.772 \\ 3.84 \end{pmatrix}, \gamma_{p2} = \begin{pmatrix} -0.200 \\ 0.996 \end{pmatrix}, \gamma_{p3} = \begin{pmatrix} 0.143 \\ -0.711 \end{pmatrix}$$

The next step is to put eq. (4-13) into the form:

$$\frac{1}{2} \Delta \mathrm{x}_p^T \mathrm{X}_p \mathrm{h}^R = \delta_{p,} \ p \in P \tag{4-34}$$

using eq. (4-21). Therefore:

$$\tilde{f}_L(x_{p1}) = -0.534, \ \tilde{f}_L(x_{p2}) = -0.278, \ \tilde{f}_L(x_{p3}) = 0.489$$

$$\delta_{p1} = -0.511, \ \delta_{p2} = -0.0852, \ \delta_{p3} = -0.170$$

Now the entire set of equations can be put in the form

$$Xh^R = b \tag{4-35}$$

where

$$\chi = \begin{bmatrix} .125 & .125 & .125 & .125 \\ .0313 & .0313 & .0313 & .0313 \\ .125 & .125 & .125 & .125 \end{bmatrix}$$

$$X = \begin{bmatrix} -.5 & -.5 & 0 & 0 \\ 0 & 0 & -.5 & -.5 \\ -.25 & -.25 & 0 & 0 \\ 0 & 0 & -.25 & -.25 \\ .5 & .5 & 0 & 0 \\ 0 & 0 & .5 & .5 \\ .125 & .125 & .125 & .125 \\ .0313 & .0313 & .0313 & .0313 \\ .125 & .125 & .125 & .125 \end{bmatrix}$$

$$b = \begin{pmatrix} -0.772 \\ 3.84 \\ -0.200 \\ 0.996 \\ 0.143 \\ -0.711 \\ -0.511 \\ -0.0852 \\ -0.170 \end{pmatrix}$$

By imposing the condition that H is symmetric, we will reduce the number of unknowns to just the symmetric parts using the linking matrix T, so that $h^R = Th^S$.

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The next step is to calculate the weighting factors using eqs. (4-27)-(4-31).

From eq. (4-29)

$$\beta_{p1} = 2, \;\; \beta_{p2} = 8, \;\; \beta_{p3} = 2$$

These are then scaled so that the smallest is equal to 1, Thus

$$\beta_{p1} = 1, \;\; \beta_{p2} = 4, \;\; \beta_{p3} = 1$$

Next, the weighting factor for each previous point is calculated using eq. (4-28).

$$W_{p1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \;\; W_{p2} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}, \;\; W_{p3} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The weighting factors to weight functions more than gradients are calculated from eqs. (4-30) and (4-31). The intermediate calculations for eq. (4-31) involve

$$W_{p1}\gamma_{p1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -0.772 \\ 3.84 \end{bmatrix} = \begin{bmatrix} -0.772 \\ 3.84 \end{bmatrix}$$

$$W_{p2}\gamma_{p2} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} -0.200 \\ 0.996 \end{bmatrix} = \begin{bmatrix} -0.800 \\ 3.98 \end{bmatrix}$$

$$W_{p3}\gamma_{p3} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.143 \\ 0.711 \end{bmatrix} = \begin{bmatrix} 0.143 \\ 0.711 \end{bmatrix}$$

leading to the weighting factor

$$w = \frac{3.98}{.0852} = 46.73$$

The terms of eq. (4-30) are

$$w\beta_{p1} = 46.73, \;\; w\beta_{p2} = 186.96, \;\; w\beta_{p3} = 46.73$$

All of these weighting factors are put together in a diagonal weighting matrix *D*.

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 46.73 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 186.96 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 46.73 \end{bmatrix}$$

We now have all the pieces to calculate the approximate Hessian from eq.(4-32).

$$h^R = \begin{bmatrix} 2.05 \\ -0.911 \\ -0.911 \\ -2.96 \end{bmatrix} \qquad H = \begin{bmatrix} 2.05 & -.911 \\ -.911 & -2.96 \end{bmatrix}$$

The actual Hessian is

$$\begin{bmatrix} 1.03 & -0.51 \\ -0.51 & -2.05 \end{bmatrix}.$$

Use of this approximate Hessian in a quadratic approximation should help the accuracy compared to a linear approximation, because the terms are of the approximate Hessian are of same order and sign as the true Hessian.

When first testing the multipoint quadratic approximation using this method to obtain the approximate Hessian, the optimizer would make large steps that would reduce the objective function and create large constraint violations. It was discovered that this was the result of the approximate Hessian sometimes having large negative terms. The optimizer would proceed in the direction to take advantage of these negative terms. This would cause the second order term to become a large negative term overwhelming the first order term. To prevent this from happening a limit was placed on the size of a negative contribution of the quadratic term in the approximation. A limit of 10 % of the size of the first order term was used and this greatly improved the accuracy of the approximation. The 10% limit worked well for the reciprocal case and 15% was used for the direct variable case. Further refinement of these limits could possibly produce further improvements in accuracy of the approximation.

4.3.2 Using Two Most Recent Points

Calculating the Hessian using the pseudo inverse method and all previous point will get quite expensive when there are many design variables and constraints. In this section another method developed by Canfield will be summarized very briefly. This method should prove to be less computationally expensive but still give a useful approximation (Canfield,1995).

As with the previous method, the elements of the function vector *f(X)* will be estimated using a quadratic Taylor series approximation of the following form:

$$\tilde{f}_Q(\mathrm{x}) = f(\mathrm{x}_k) + [\nabla \mathrm{f}(\mathrm{x}_k)]^T \Delta\mathrm{x} + \frac{1}{2}\Delta\mathrm{x}^T \mathrm{H}\Delta\mathrm{x} \qquad \text{(4-36)}$$

Where x_k is the design vector at the *k*th iteration, $\Delta\mathrm{x} = \mathrm{x}_k - \mathrm{x}_{k-1}$, is the change in the design vector, $\nabla\mathrm{f}(\mathrm{x}_k)$ is the function gradient, and H is an approximation to its Hessian. Again, if H were composed of analytic second derivatives, this would represent a second order Taylor series approximation.

Consider updating the Hessian based on the two most recent design points

$$H^* = H_{k-1} + U_k \qquad \text{(4-37)}$$

where U_k is chosen to satisfy the condition

$$H^*\{x_{k-1} - x_k\} = \nabla f(x_{k-1}) - \nabla f(x_k) \qquad (4\text{-}38)$$

One solution for U_k is an update matrix based on Powell's Modified BFGS rank two update

$$U_k = \frac{qq^T}{q^T s} - \frac{H_k s s^T H_k}{s^T H_k s} \qquad (4\text{-}39)$$

where

$$q = \theta\gamma + (1-\theta)H_k s$$

$$s \equiv x_k - x_{k-1}$$

$$\theta = \begin{cases} 1 & \text{if } s^T\gamma \geq 0.2 H_k s; \\ \dfrac{0.8 s^T H_k s}{s^T H_k s - s^T \gamma} & \text{otherwise} \end{cases}$$

The Hessian is further updated using the following update formula to match the previous function value.

$$H_k = H^* + U_k^* = H_{k-1} + U_k + U_k^* \qquad (4\text{-}40)$$

where

$$U_k^* = \frac{2\left[f(x_{k-1}) - f(x_k) - \{\nabla f(x_k)\}^T s - \frac{1}{2} s^T H^* s\right]}{(s^T s)^2} s s^T \qquad (4\text{-}41)$$

H_K is the new Hessian approximation.

4.4 Spherical Move Limit

It is convenient to apply a spherical move limit because it is also a quadratic constraint. Numerical experiments here have shown that convergence seems to be less sensitive to spherical move limits than the conventional infinity norm side constraints in which the approximation is limited to a hypercubical region. The spherical move limit is posed as

$$(x - x_k)^T S_k (x - x_k) - r_k \leq 0 \tag{4-42}$$

where the terms of the diagonal scaling matrix S_k are given by

$$S_{ij}^{(k)} = \frac{\delta_{ij}}{\sqrt{x_i^{(k)}}}, \quad i = 1,2,\ldots,n \tag{4-43}$$

at the *k*th iteration for *n* design variables. The radius of the sphere r_k may vary from one iteration to the next; however, the use of quadratic constraint approximations can decrease the need to gradually reduce it, as is often required for first order approximations to converge.

For most of the cases the initial move limit is applied and a reduction factor is used after each iteration to decrease the move limit as the approximation gets closer to the optimum. This helps convergence by not letting the approximation go so far that inaccuracies keep it away from the optimum. This reduction factor can apply to any type of move limit, spherical or box.

5. Numerical Results

5.1 Overview

The purpose of this research is to develop a more efficient method of structural optimization which will garner the higher accuracy and quicker convergence of a quadratic Taylor series approximation without incurring the expense of calculating the Hessian matrix. The approach involves fully utilizing the information from previous iterations. All the exact structural and sensitivity analyses will be used to estimate the Hessian matrix for a quadratic approximation to the response quantities.

Once this method was programmed, it was tested using several known structural examples from literature: a five section beam, a four member frame, a ten bar truss, a twelve member frame, a 113 member truss called ACOSS II linked to 6 design variables and 31 design variables, and finally a 40 member frame. These seven structures have a wide range of constraints including stress, displacement and frequency constraints. These structures were optimized using eight different techniques.

The techniques are broken into two categories: direct and reciprocal variables. The first four techniques used direct variables which are the cross sectional areas of the members. The first of the direct methods used is the optimization routine in MATLAB called "constr", written for constrained optimization (Grace,1993:9-11). It is given the exact problem to solve. This method is also used by all the others to solve the approximate problems. The second direct method used was MQA-D which stands for Multipoint Quadratic Approximation in Direct variables. This method is an implementation of the equations in section 4.3.1 used to generate the approximate Hessian. The third method, SPA-DB, is a single point linear approximation, identical to Sequential Linear

Programming (SLP). The fourth direct method was SPA-DS which is the same as SPA-DB except it uses spherical move limits.

There were four reciprocal variable methods used for comparison. The first is MQA-R, which is similar to MQA-D above except that reciprocal intermediate design variables are used. It is compared to three other methods; the first, SPA-RS, is a single point approximation in reciprocal variables, with spherical move limits discussed in the section 4.3. The next method, SPA-RB, is the same as SPA-RS except that box type move limits are used. The last method, SQA-R, is sequential quadratic approximation designed to be less computationally expensive than MQA-R. It is an implementation of the equations in section 4.3.2.

The structural analyses for the 4-Member Frame, 12-Member Frame, and 40 Member Frame are done by a program called FRAMIN (Kolonay,1987). The ten bar truss and both the ACOSS models are analyzed by a program called SALVO (Canfield,1986). The optimization routine is written in MATLAB and interfaces with SALVO and FRAMIN.

CPU times were measured for some of the problems. It must be empasized that this was done just to get a preliminary idea of the expense of the different approximations. It is understood that estimating the Hessian using the pseudo inverse can get computationally expensive. The intent was to get an idea of how large a problem would make the computational expense unbearable compared to the expense of the finite element analysis. None of the structures optimized in the research are computaionally expensive to analyze. Therefore, it is doubtful that the quadratic methods would require less CPU time. The purpose is to see if the quadratic methods can optimize the structures in fewer iterations and to see if the CPU times are reasonable. If the quadratic method can optimize a structure in fewer iterations then perhaps it will be more economical for a certain range of problems with a small number of design variables, but that require a large amount of CPU time to analyze. Another reason why the CPU times should only be used for a preliminary look is that the finite element analyses are all in FORTRAN which is very fast compared to

MATLAB in which the optimization routines were written. Again, the CPU times should only be compared to see if the method is in the ballpark.

5.2 Comparison of Methods

5.2.1 Five Section Beam

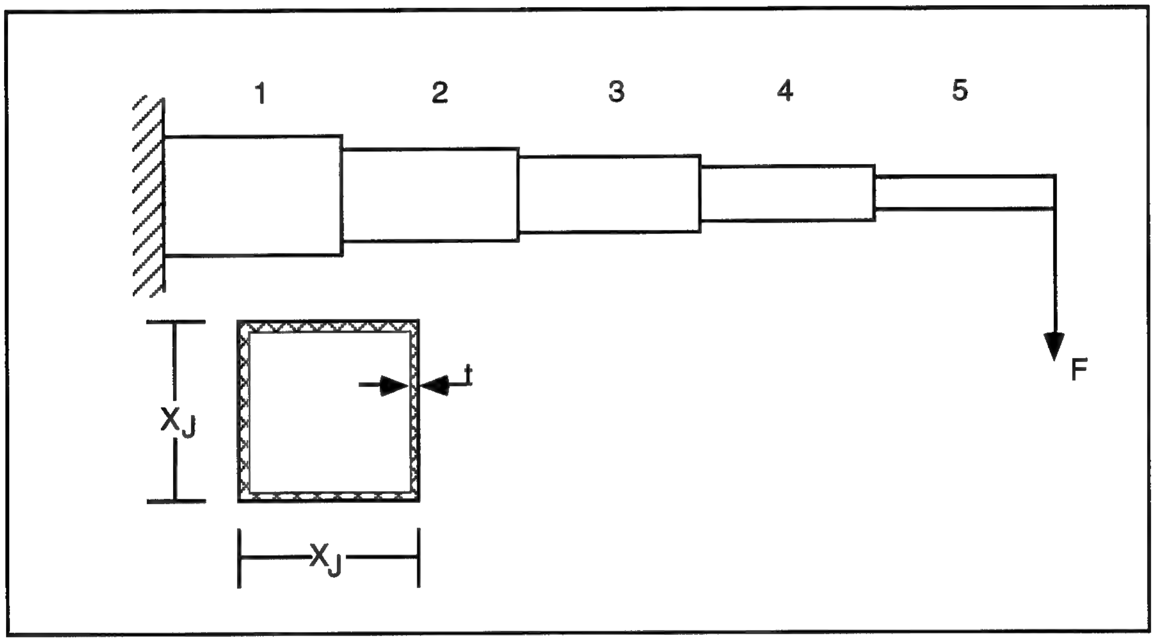


Figure 5-1: 5-Section Beam

5.2.1.1 Description

The first model optimized is a 5 section cantilever beam obtained from a paper by Toropov, Filatov, and Polynkin; see Figure 5-1 (Toropov,1993,7-14). The beam is a square box section with five different cross sections. The thickness of each section is uniform while the design variable x_j controls the height and width of the *j*th section. The constraint is the tip displacement. The objective function *f(X)* and constraint function *g(X)* are:

$$f(X) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5)$$

$$g(X) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$$

A feasible starting point X_0={5,5,5,5,5} was chosen where $f(X_0)$=1.560 and $g(X_0)$=0.0. Toropov *et al.* converged to the optimized objective function of 1.34 in nine iterations using convergence criteria of the change in objective function of $\leq$ 0.01 and maximum constraint violation of 0.02.

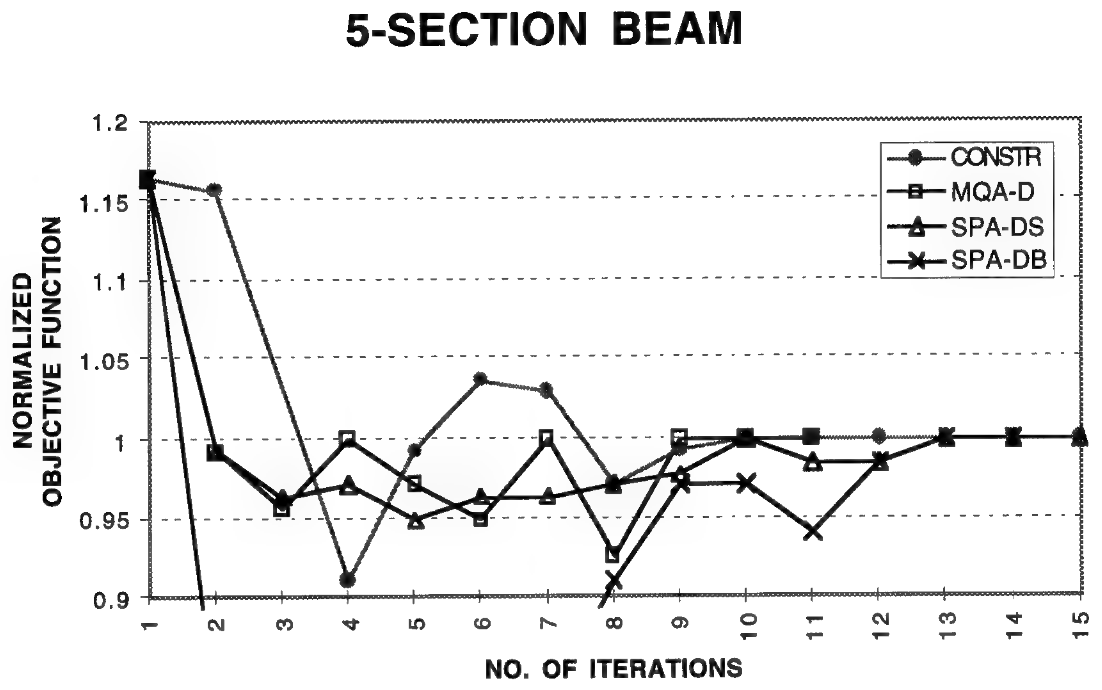
5.2.1.2 Results

5.2.1.2.1 Optimization Problem Overview

Table 5-1 gives an overview of the information used in this optimization problem. All of the methods converged to the same optimum.

Table 5-1 5-Section Beam Optimization Overview

No. of Design Variables	5
No. of constraints (type)	1(Disp)
Convergence Criteria	
Max. Change in Objective Function	.001
Max. Constraint Violation	.001
Max. Change in Design Variable	.01
Move Limits	.5 with .9 reduction factor
Min. Allowable Design Variable	.01
Max. Allowable Design Variable	10
Initial Objective Function	1.56
Initial Design Variables	5,5,5,5,5
Initial Max. Constraint Violation	0
Optimum Objective Function	1.34
Optimum Design Variable Values	6.05,5.27,4.47,3.53,2.15



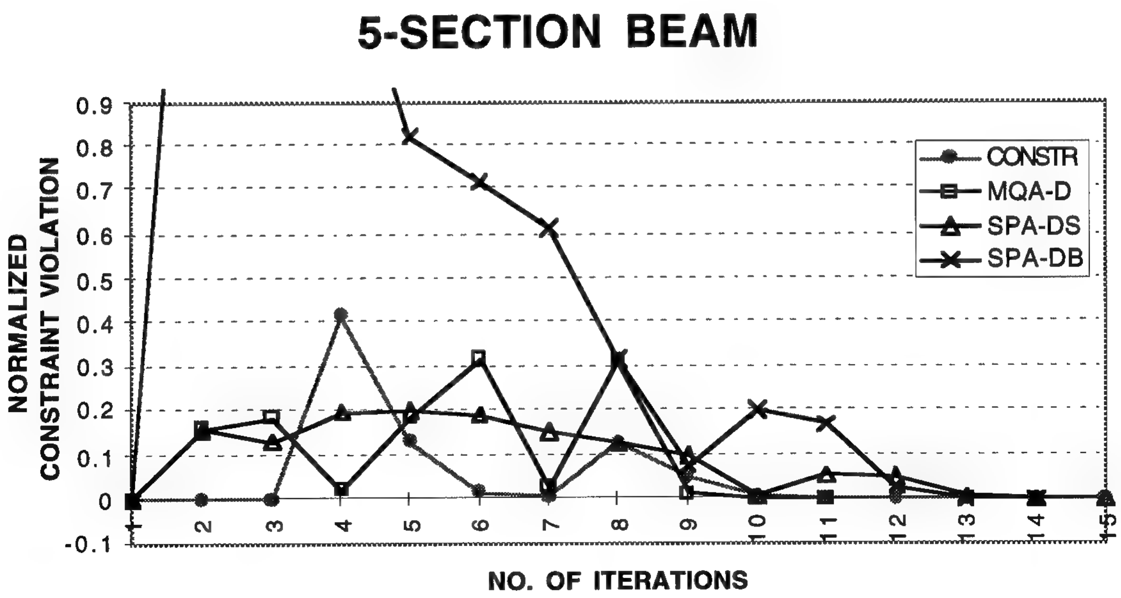


Figure 5-2: Direct Method Optimization

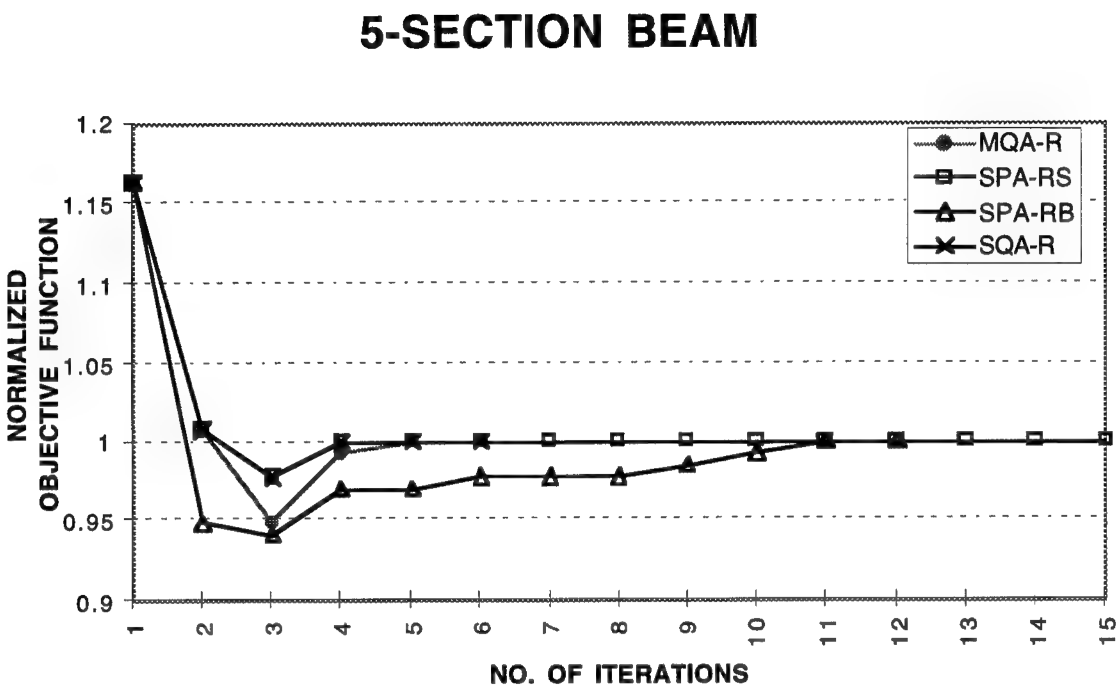
5.2.1.2.2 Direct Variable Optimization

Figure 5-2 shows the results for the four methods using direct design variables. Table 5-2 shows the exact number of iterations each method took to reach the convergence criteria. This chart also shows the CPU time each method required to converge.

Table 5-2: Direct Method Convergence

Method	No. Of Iterations to Converge	Time In CPU Sec
constr	16	4
MQA-D	11	50
SPA-DS	20	39
SPA-DB	14	13

Since this is a very simple problem, solving the exact problem is very quick. It is very reasonable for 'constr' to solve this problem very quickly in CPU seconds. However, if this were not the case and it took much longer to solve the exact function and gradients then MQA-D has a considerable advantage. MQA-D is intended to be more CPU efficient for problems where the analysis is very expensive which is definitely not the case for this problem. The purpose here is to show that it can solve the problem in fewer iterations so that computational savings could be achieved if the analysis were very expensive. All of these methods converged in spite of oscillation as they approached the optimum.



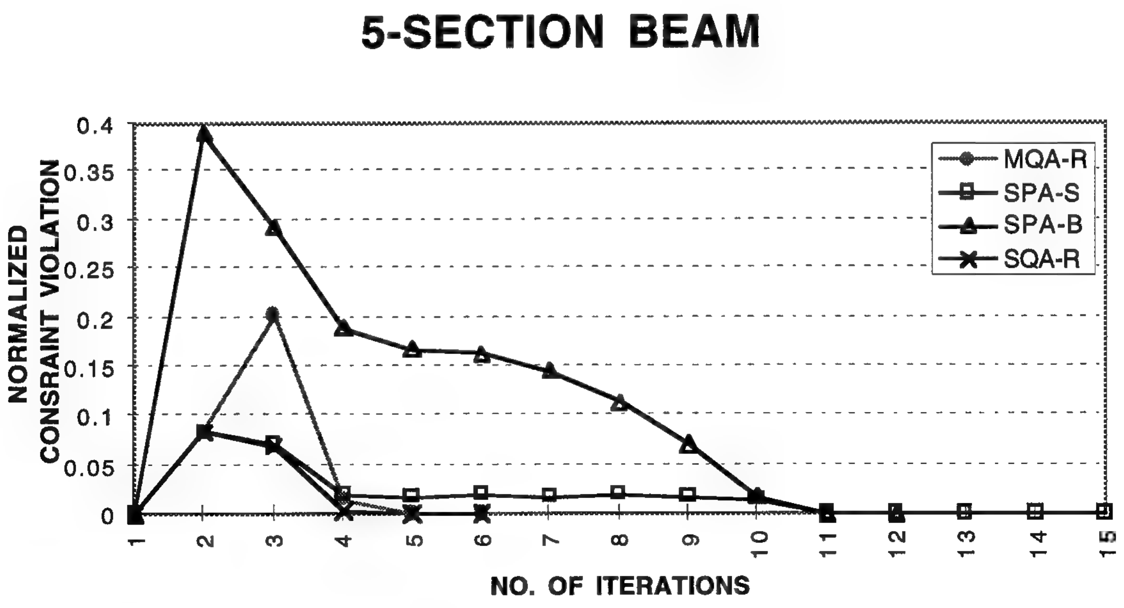


Figure 5-3: Reciprocal Method Optimization

5.2.1.2.3 Reciprocal Variable Optimization

Table 5-3 shows the comparison of the four reciprocal methods. Table 5-3 gives the exact number of iterations to converge to the required tolerance and the CPU time.

Table 5-3: Reciprocal Method Convergence

Method	No. Of Iterations to Converge	Time In CPU Sec
MQA-R	6	6
SPA-RS	16	11
SPA-RB	12	11
SQA-R	6	4

The two quadratic methods are very comparable in solving the problem and solve it very quickly. SPA-RS gets very close to the answer in four iterations but still takes another 12 to meet the criteria. If the criteria had been less stringent it would have done almost as well as the quadratic methods. The apparent benefit of the quadratic approximations is their ability to converge without sensitivity to move limits. In contrast, the single point methods cannot converge to the required tolerance until the move limits are reduced enough to be compatible with the termination criteria. SPA-RB Converged faster than SPA-RS but it was not closer to the optimum until 10 iterations.

5.2.1.2.4 Comments

For this problem the quadratic methods definitely helped speed convergence as far as number of iterations, and for the reciprocal case it is faster even for this small problem. Even though the box move limit single point approximation, SPA-RB converged in fewer iterations than the spherical move limit SPA-RS, it took a long route to get there. With minor adjustments in the move limits and/or convergence criteria the spherical move limits would be superior. It is interesting to note that MQA-D took almost five times as long to solve the problem in CPU seconds but only twice the number of iterations as MQA-R. This is because as more points are accumulated it takes each iteration longer to calculate the approximate Hessian. The CPU times are not considered significant for this explicit problem, which did not involve any finite element analysis. Rather CPU times were measured to verify the approach for a problem with an analytical solution.

5.2.2 Four Member Frame

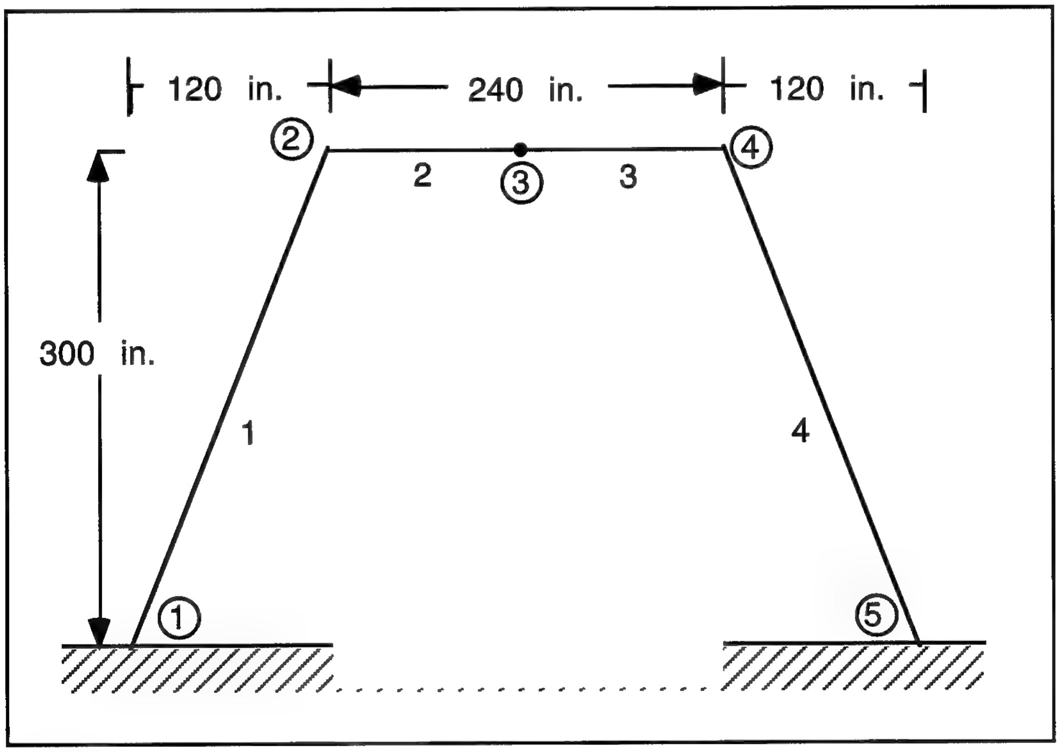


Figure 5-4: 4-Member Frame

5.2.2.1 Description

The four-member, single bay, single story frame with I cross section in Figure 5-4 was subjected to three load cases as shown in Table 5-4. The structure has 5 nodes and was subject to 45 displacement constraints and 12 stress constraints for the three load cases shown in Table 5-2. For this structure there is a fixed relationship defined between the moment of inertia(I), section modulus (S) and the cross sectional area (A): I=0.20720xA^3, S=0.39300xA^2. Where the constant, 0.20720, has units of in^{-2} and the constant, 0.39300, has units of in^{-1}. This model was obtained from Ray M. Kolonay, Wright Labs, WPAFB, Ohio. Table 5-5 and Table 5-6 give material and element properties.

Table 5-4: 4-Member Load Cases

		Direction of load	
Load Condition	Node	x	y
1	2		-10000
	3		-10000
	4		-10000
2	2	5000	-10000
	3		-10000
	4		-10000
3	2		-10000
	3		-10000
	4	-5000	-10000

Table 5-5: 4-Member Frame Material Properties

Material	Steel
Modulus of Elasticity	29x10^6 psi
Specific Weight	.283 lb/in^2
Upper Limit on Area	88.0 in^2
Lower Limit on Area	.5 in^2
Displacement limits	x(.3 in^2),y(.15in^2),z(50.0rad)
Number of Loading Conditions	3

Table 5-6: 4-Member Element Properties

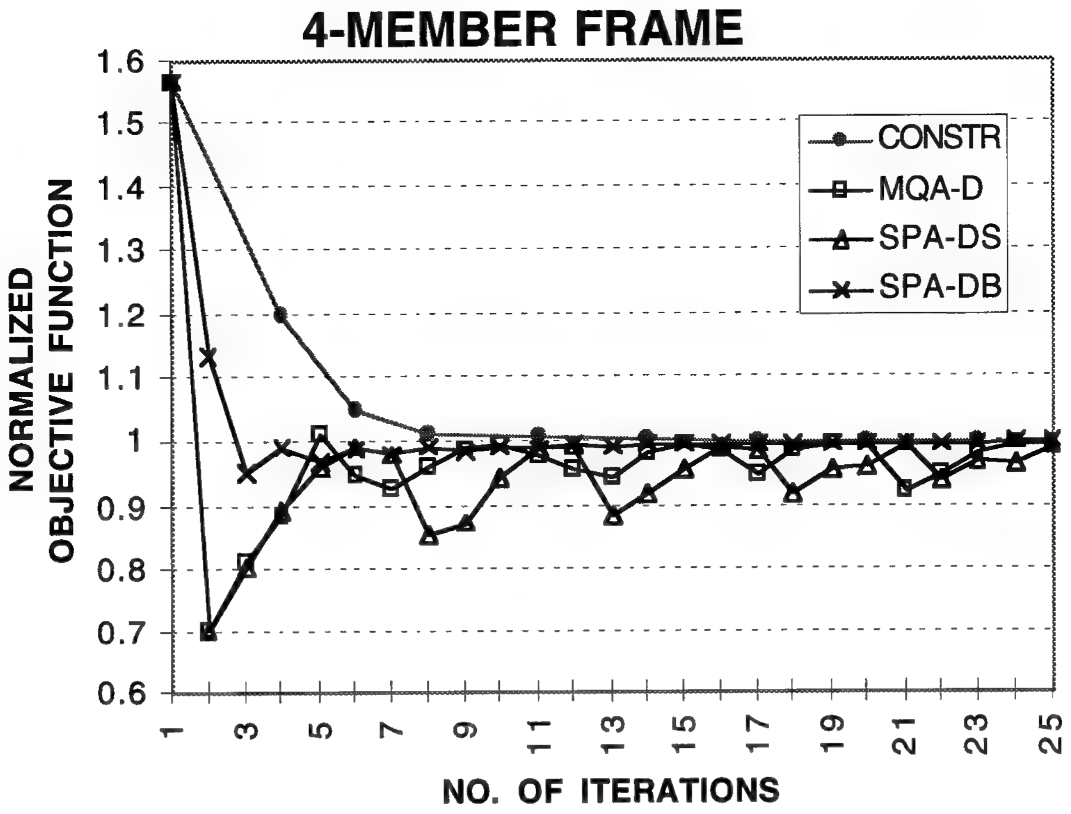
Area	20 in^2
Section Modulus	157.2 in^3
Moment of Inertia	1657 in^4
Maximum Allowable Stress	59000 psi
Initial Weight	5016.0 lb

5.2.2.1.1 Optimization Problem Overview

Table 5-7 gives an overview of the information used in this optimization problem. All of the methods converged to the same optimum.

Table 5-7: 4-Member Optimization Overview

No. of Design Variables	4
No. of constraints (type)	45(Disp), 12(Stress)
Convergence Criteria	
Max. Change in Objective Function	.1
Max. Constraint Violation	.001
Max. Change in Design Variable	.1
Move Limits	1 with .97 reduction
Min. Allowable Design Variable	.5
Max. Allowable Design Variable	88
Initial Objective Function	5016
Initial Max. Constraint Violation	-.702
Optimum Objective Function	3204
Optimum Design Variable Values	12.72,12.92,12.92,12.72



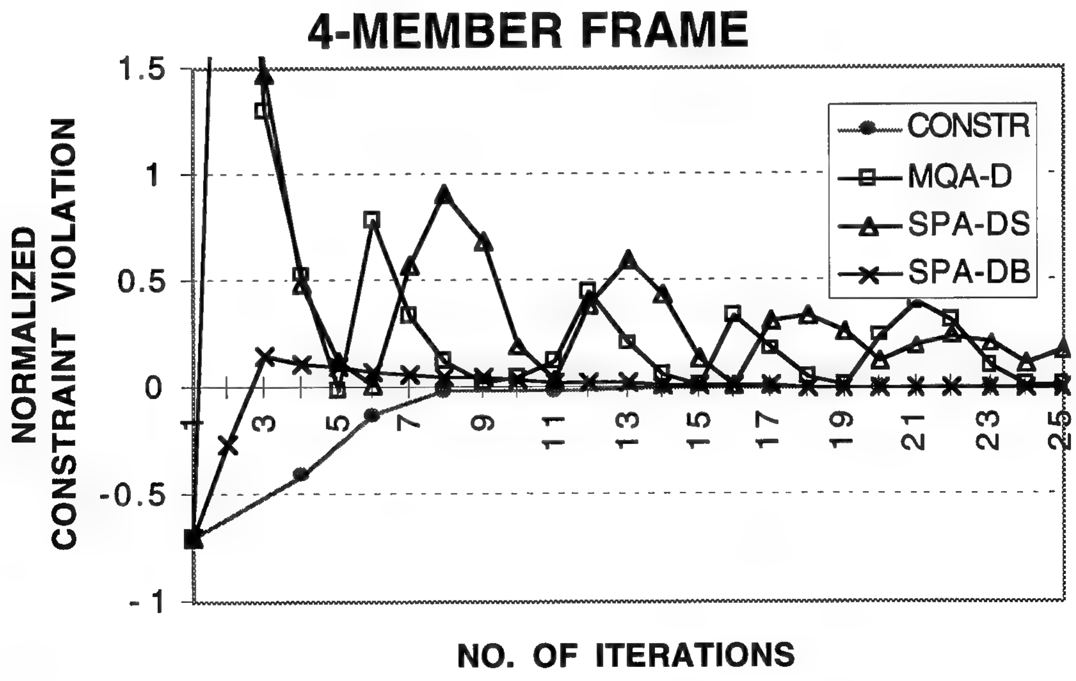


Figure 5-5: Direct Variable Optimization

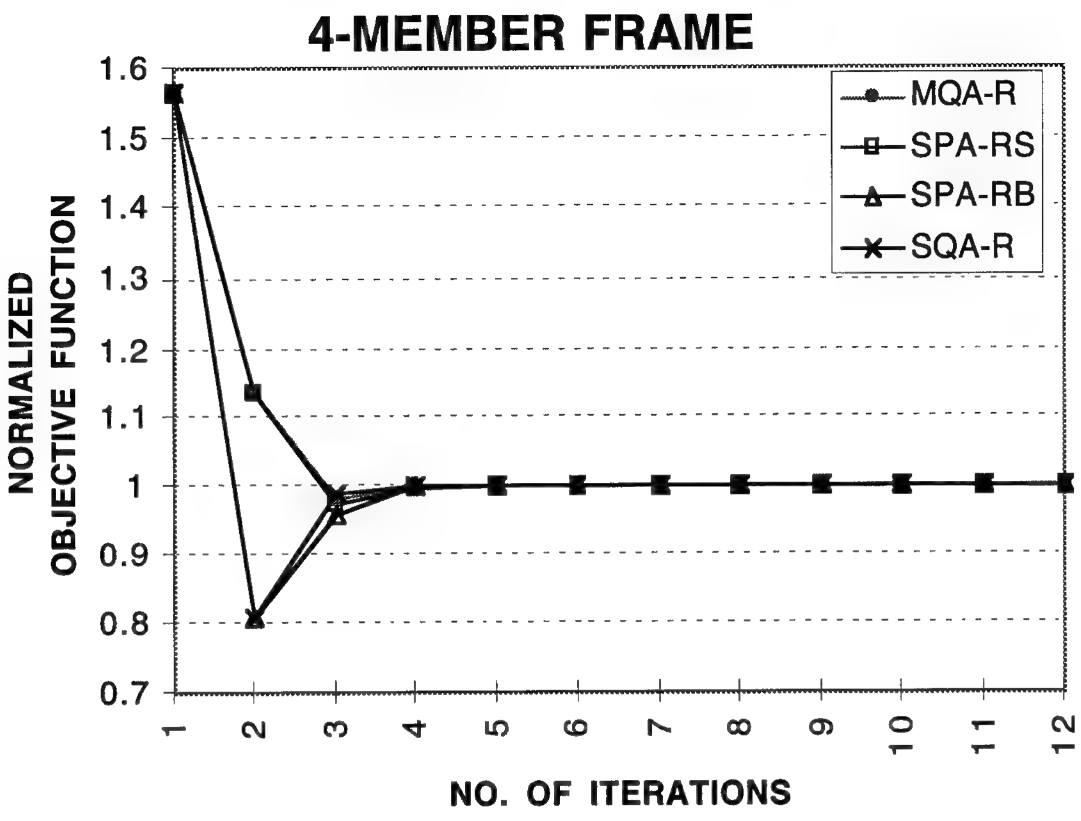
5.2.2.1.2 Direct Variable Optimization

Figure 5-5 shows the results for the three methods using direct design variables. Table 5-8 shows the exact number of iterations each method took to reach the convergence criteria.

Table 5-8: Direct Method Convergence

Method	No. Of Iterations to Converge
constr	47
MQA-D	38
SPA-DS	116+
SPA-DB	40

SPA-DS was very close to the optimum at 116 iterations. It met the change in objective function and the maximum constraint violation criteria, but it did not quite meet the change in the design variable criteria when it was stopped. The quadratic method solved the problem in the fewest number of iterations, but did not have a tremendous advantage over 'constr'.



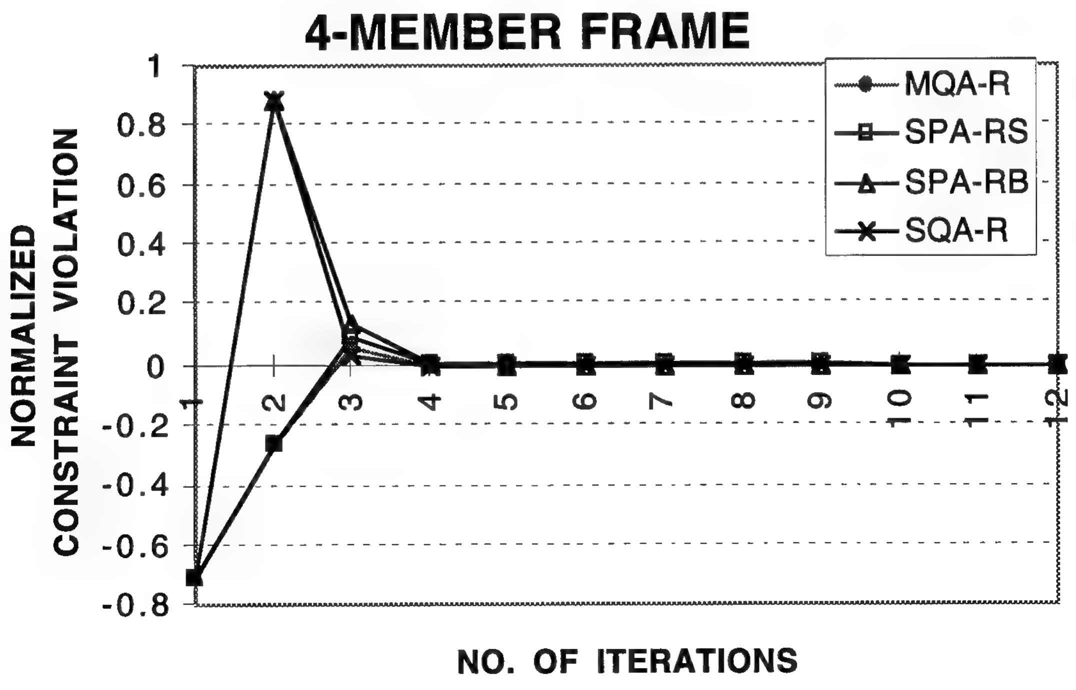


Figure 5-6: 4-Member Frame Reciprocal Method Optimization

5.2.2.1.3 Reciprocal Variable Optimization

Figure 5-6 shows the comparison of the four reciprocal methods. Table 5-9 gives the exact number of iterations to converge to the required tolerance.

Table 5-9: 4-Member Frame Convergence

Method	No. Of Iterations to Converge
MQA-R	5
SPA-RS	14
SPA-RB	12
SQA-R	12

All of the reciprocal methods came very close to the optimum in about 4 or 5 iterations; however, the single point methods took longer to converge within tolerance. The evident advantage of the MQA-R was its capability to converge cleanly without tightening of the move limits, whereas the other methods needed help from the tightened move limits to converge. For this problem the reciprocal methods showed a tremendous advantage over the direct methods.

5.2.3 Ten Bar Truss

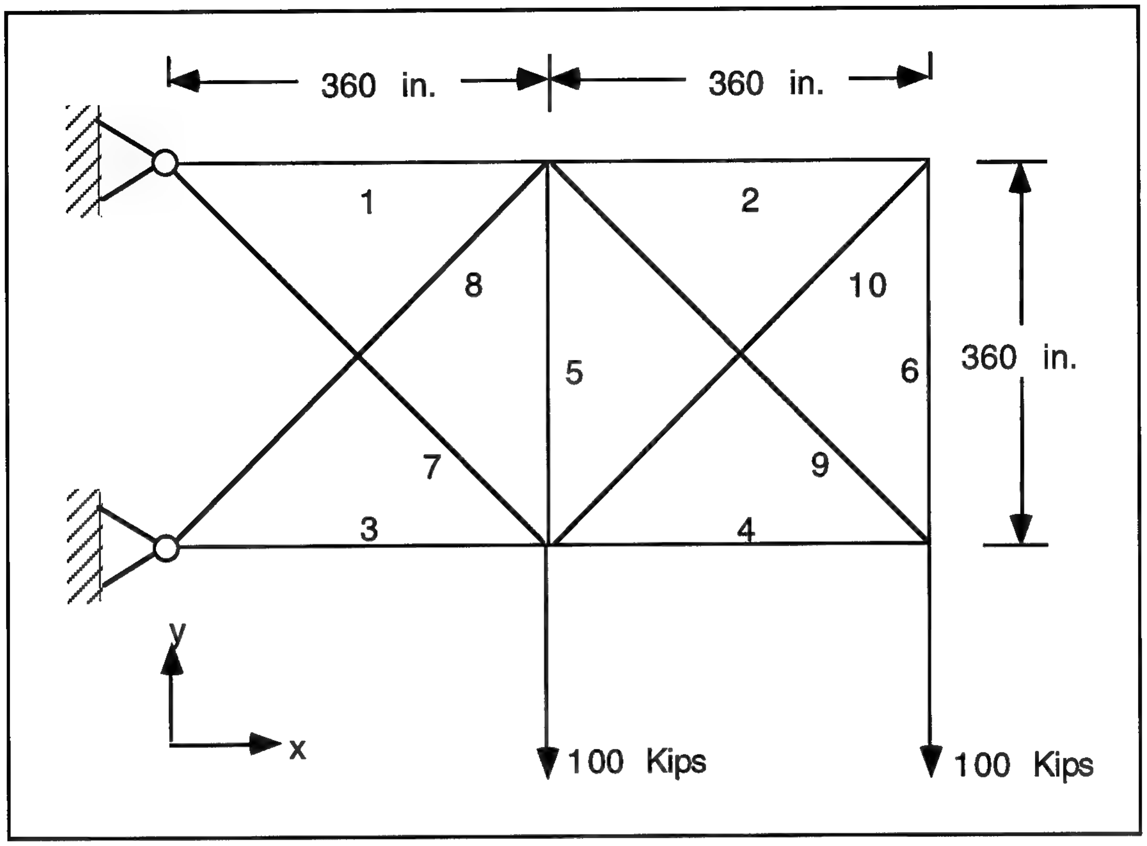


Figure 5-7: Ten Bar Truss

5.2.3.1 Description

The ten bar truss shown in Figure 5-7 is a classic example used to test optimization routines (Haftka,1992:238-239). It has ten members, six nodes, and the allowable stress is 25,000 psi in tension or compression. The area of each member was initially 10.0 in^2. At node 2 and 4 there is a 100,000 lb load in the negative y direction. The structure was restrained in the z-direction.

5.2.3.2 Results

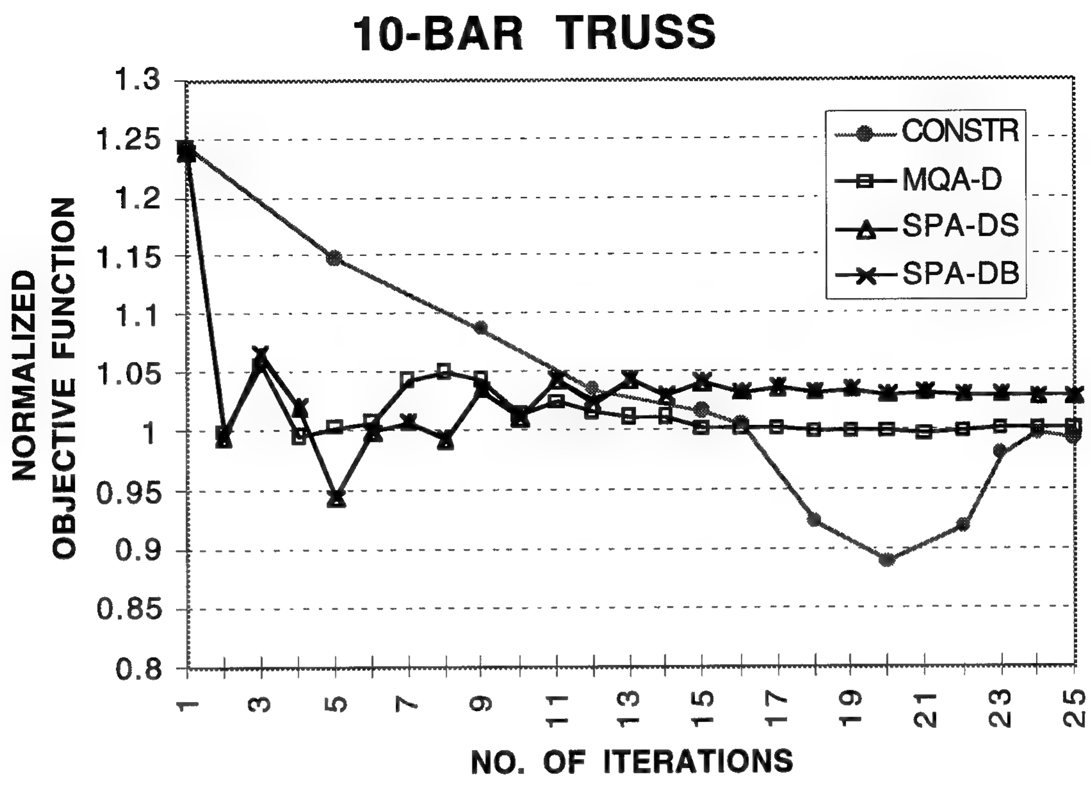
5.2.3.2.1 Optimization Problem Overview

This problem has an interesting occurrence in that there are two distinct local optima very close to each other. Any nonlinear problem can have more than one minimum. For this thesis either is considered correct, because all of the optimization techniques available, only have the capability of finding a local minimum.

Table 5-10: Ten Bar Truss Optimization Overview

No. of Design Variables	10
No. of constraints (type)	14(Disp)
Convergence Criteria	
Max. Change in Objective Function	1
Max. Constraint Violation	.001
Max. Change in Design Variable	.1
Move Limits	1.5 with .9 reduction for direct 2.0 with no reduction for recip
Min. Allowable Design Variable	.1
Max. Allowable Design Variable	40
Initial Objective Function	6294.7
Initial Design Variables	20,20,20,20
Initial Max. Constraint Violation	.3132
Optimum Objective Function	5060.9, 5076.7 *
Optimum Design Variable Values	[30.52, .1, 22.20, 15.22, .1, .56, 7.46, 21.03, 21.53, .1] [30.73,.1,23.94,14.73,.1, .1,8.54,20.95,20.84,.1]*

* Two different optima.



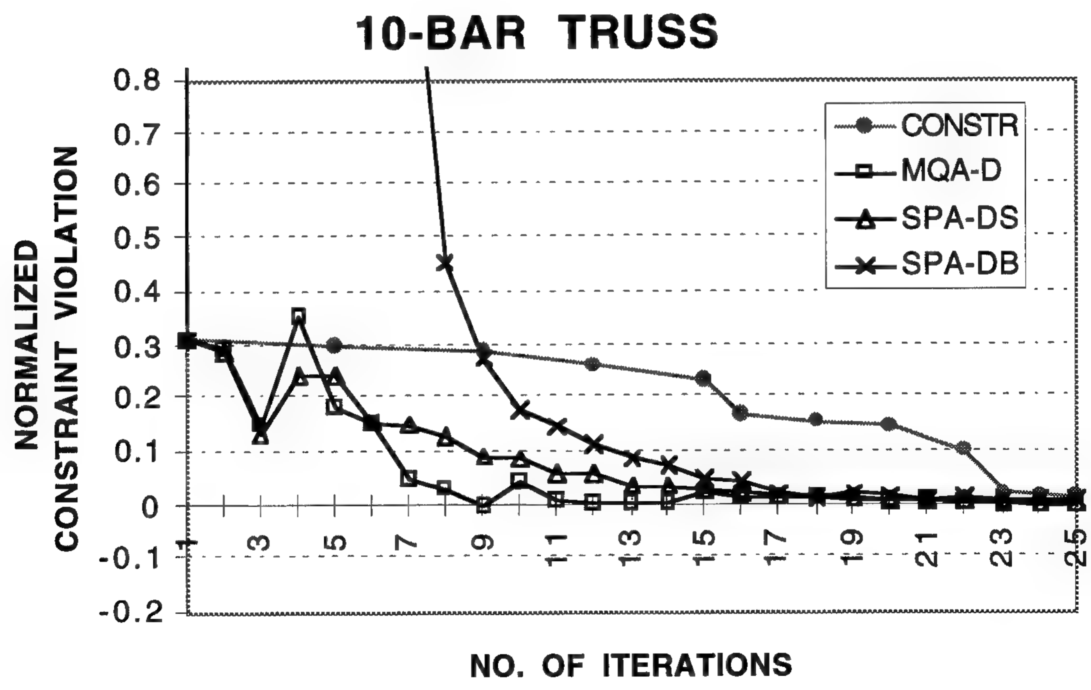


Figure 5-8: Ten Bar Truss Direct Optimization

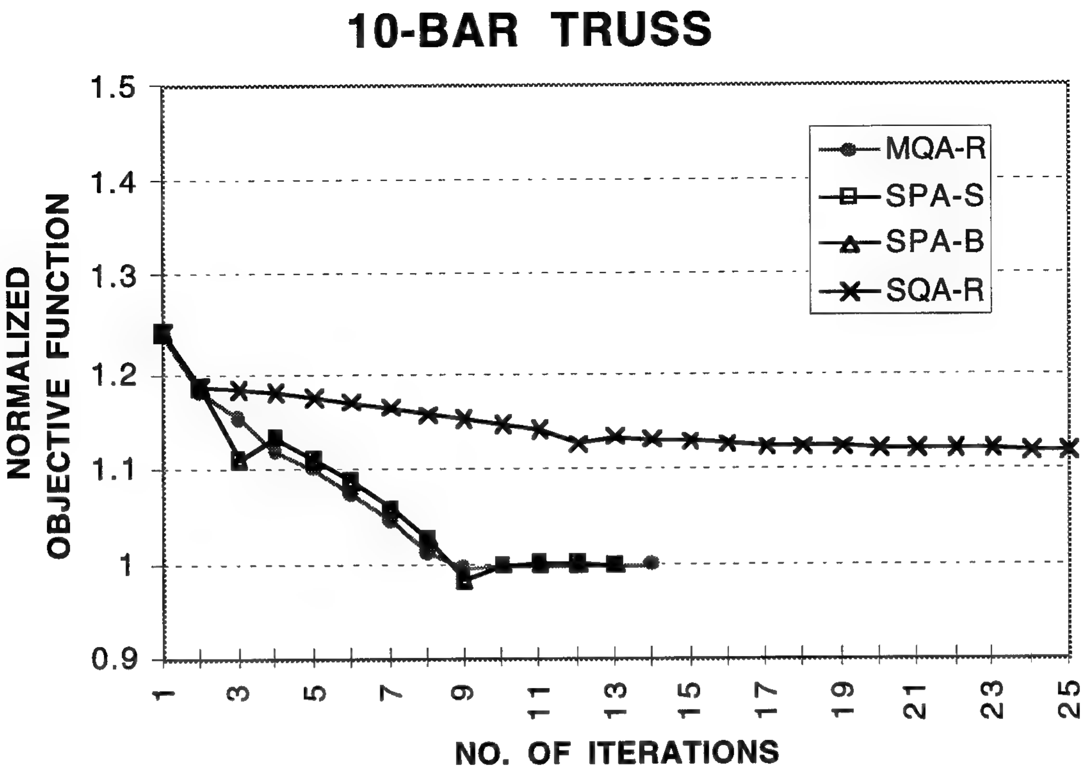
5.2.3.2.2 Direct Variable Optimization

Figure 5-8 shows the results for the three methods using direct design variables. Also, shown in Table 5-11 are the exact number of iterations each method took to reach the convergence criteria.

Table 5-11: Ten Bar Truss Convergence

Method	No. Of Iterations to Converge
constr	34
MQA-D	35
SPA-DS	50+
SPA-DB	73+

For this problem 'constr' and MQA-D are very comparable in final efficiency; however, MQA-D appears to be very close to a reasonable engineering solution in about 20 iterations. SPA-DS and SPA-DB have a great deal of trouble because of the move limits. The move limit reduction factor closes in on SPA-DS and SPA-DB before they are very close to the optimum, so they get bound from reaching it. This problem is very sensitive to the move limits in the direct variables. The same move limit as the reciprocal approach could not be used because none of the direct methods would even come close to converging. SPA-DS will converge in about 50 iterations if the move limits are reduced at a slower rate, such as 0.95.



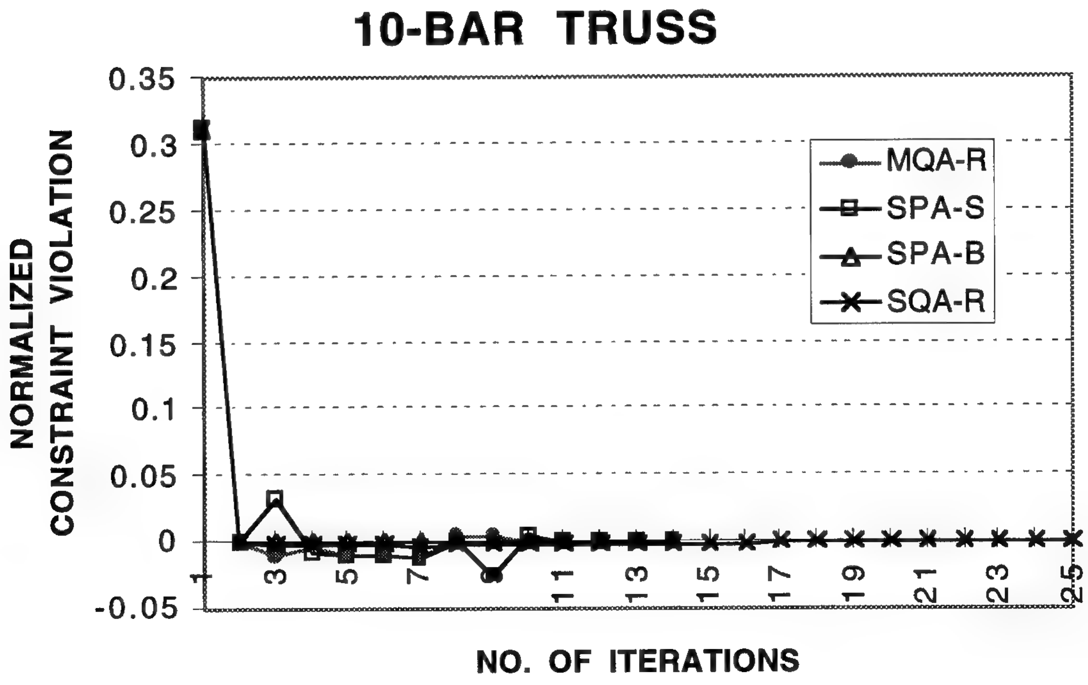


Figure 5-9: Ten Bar Truss Optimization

5.2.3.2.3 Reciprocal Variable Optimization

Figure 5-9 shows the comparison of the four reciprocal methods. Table 5-12 gives the exact number of iterations to converge to the required tolerance.

Table 5-12: Ten Bar Truss Convergence

Method	No. Of Iterations to Converge
MQA-R	14
SPA-RS	14
SPA-RB	14
SQA-R	36+

A large move limit was used for this problem so that the accuracy of the reciprocal approaches could be tested. MQA-R, SPA-RS, and SPA-RB solve the problem in about ten iterations and then took about three more to reach the criteria. SQA-R had trouble with this problem. It would eventually converge in about 150 iterations. The problem here seems to be that SQA-R uses a Hessian approximation that is positive definite. This causes this approach to be very conservative if the true Hessian is not close to positive definite. I believe that this is what is happening here.

The reciprocal approaches for this problem have an advantage over the direct in that a move limit is almost not necessary for the reciprocal. Also the reciprocal methods converged much faster when getting close to the optimum. For this problem the quadratic terms in reciprocal approximations do not speed convergence.

5.2.4 12 Member Frame

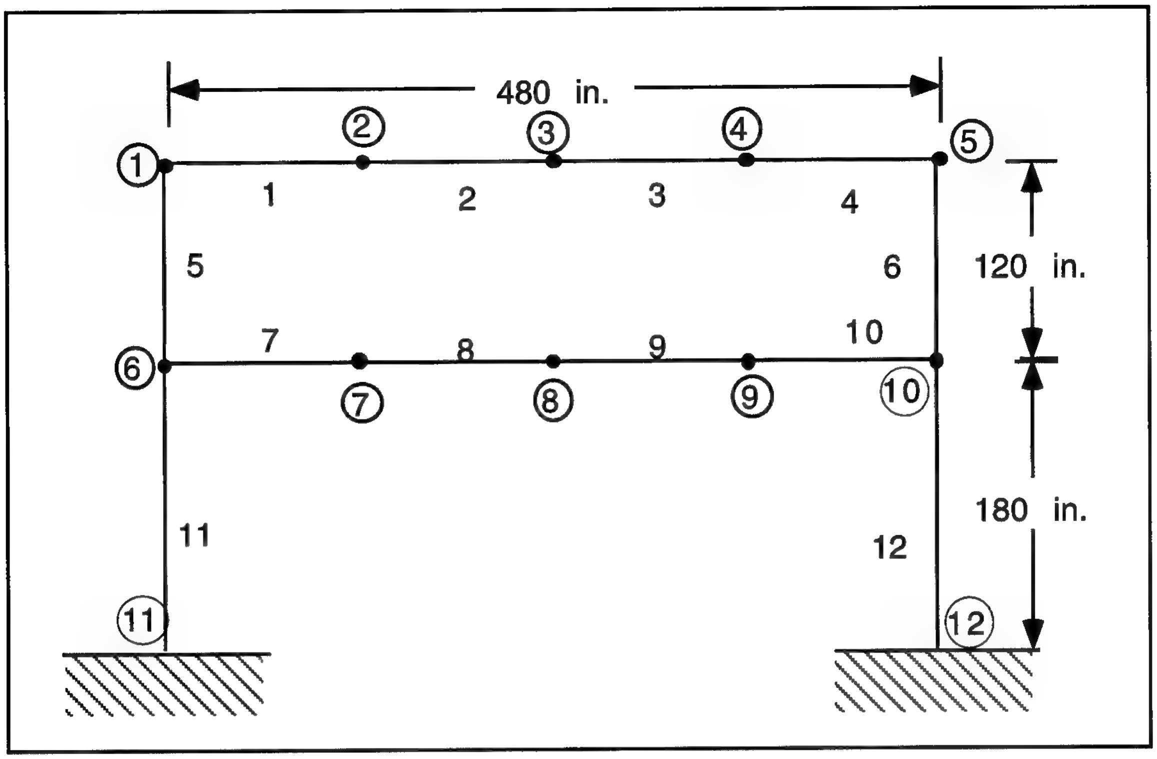


Figure 5-10: 12 Member Frame

5.2.4.1 Description

The twelve member, two story, single bay frame with I-section elements in Figure 5-10 was subjected to three load cases shown in Table 5-13, stress constraints on each element and limited nodal displacement in the x-direction. There are 12 nodes in the structure. Nodes 11 and 12 are restrained from displacement. For this structure there is a fixed relationship between the moment of inertia(I), section modulus (S) and the cross sectional area (A): $I=0.20720xA^3$, $S=0.39300xA^2$ (Kolonay,1987,92), where the constant, 0.20720, has units of in^{-2} and the constant, 0.39300, has units of in^{-1}. Table 5-14 and Table 5-15 give element and member properties.

Table 5-13: 12 Member Frame Load Cases

		Direction of load	
Load Case	Node	x(lb)	y(lb)
1	1		-5000
	2		-10000
	3		-10000
	4		-10000
	5		-5000
	6		-10000
	7		-20000
	8		-20000
	9		-20000
	10		-10000
2	1	1785	-5000
	2		-10000
	3		-10000
	4		-10000
	5	1785	-5000
	6	4460	-10000
	7		-20000
	8		-20000
	9		-20000
	10	4460	-10000
3	1	-1785	-5000
	2		-10000
	3		-10000
	4		-10000
	5	-1785	-5000
	6	-4460	-10000
	7		-20000
	8		-20000
	9		-20000
	10	-4460	-10000

Table 5-14: 12 Member Frame Material Properties

Material	Steel
Modulus of Elasticity	29×10^6 psi
Specific Weight	.283 lb/in^2
Upper Limit on Area	88.0 in^2
Lower Limit on Area	.5 in^2
Number of Loading Conditions	3
Displacement limits	
Node 1,6	x(.3 in^2)
Node 3,8	y(.9 in^2)

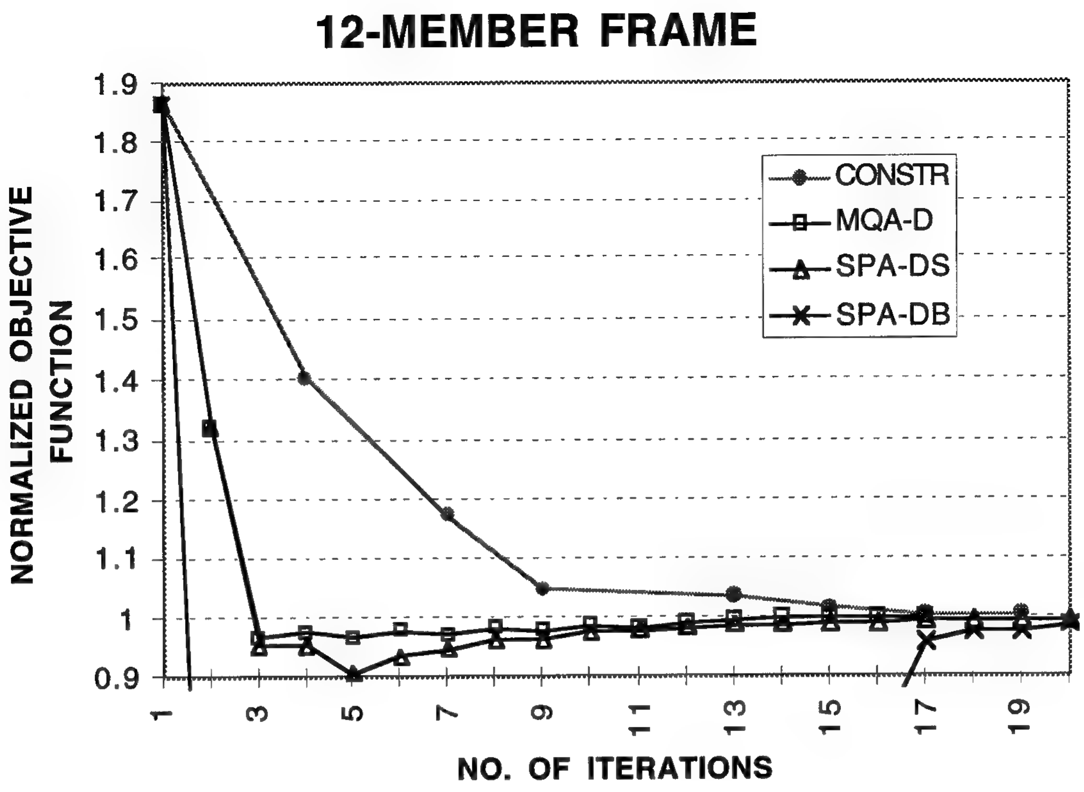
Table 5-15: Element Properties

Area	30 in^2
Section Modulus	353.7 in^3
Moment of Inertia	5594 in^4
Maximum Allowable Stress	129000 psi
Initial Weight	13244.4 lb

Table 5-16 gives an overview of the information used in this optimization problem. All of the methods converged to the same optimum.

Table 5-16: 12 Member Frame Optimization Overview

No. of Design Variables	12
No. of constraints (type)	12(Disp), 36(Stress)
Convergence Criteria	
Max. Change in Objective Function	1
Max. Constraint Violation	.001
Max. Change in Design Variable	.1
Move Limits	1 with .9 reduction
Min. Allowable Design Variable	.5
Max. Allowable Design Variable	88
Initial Objective Function	13244
Initial Design Variables	30,30,30,30,30,30,30,30,30,30,30,30
Initial Max. Constraint Violation	-.787
Optimum Objective Function	7085
Optimum Design Variable Values	14.15, 14.00, 13.94, 14.18, 14.50, 21.15, 16.72,16.50,21.21,16.18,16.03



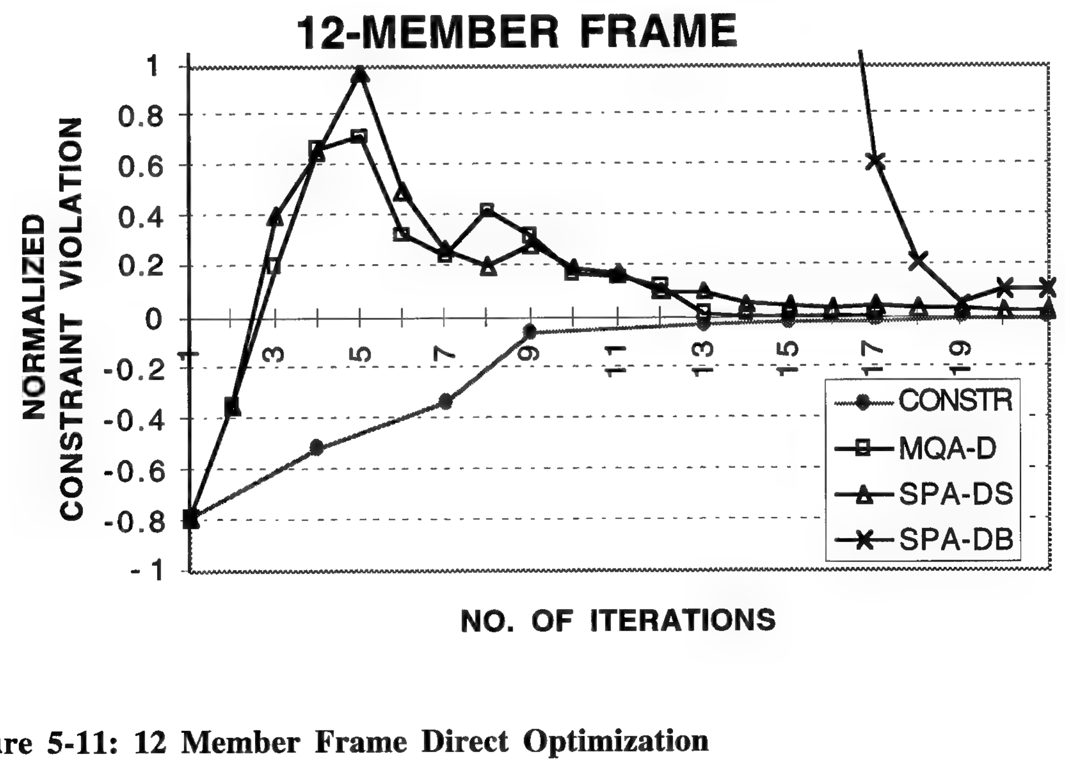


Figure 5-11: 12 Member Frame Direct Optimization

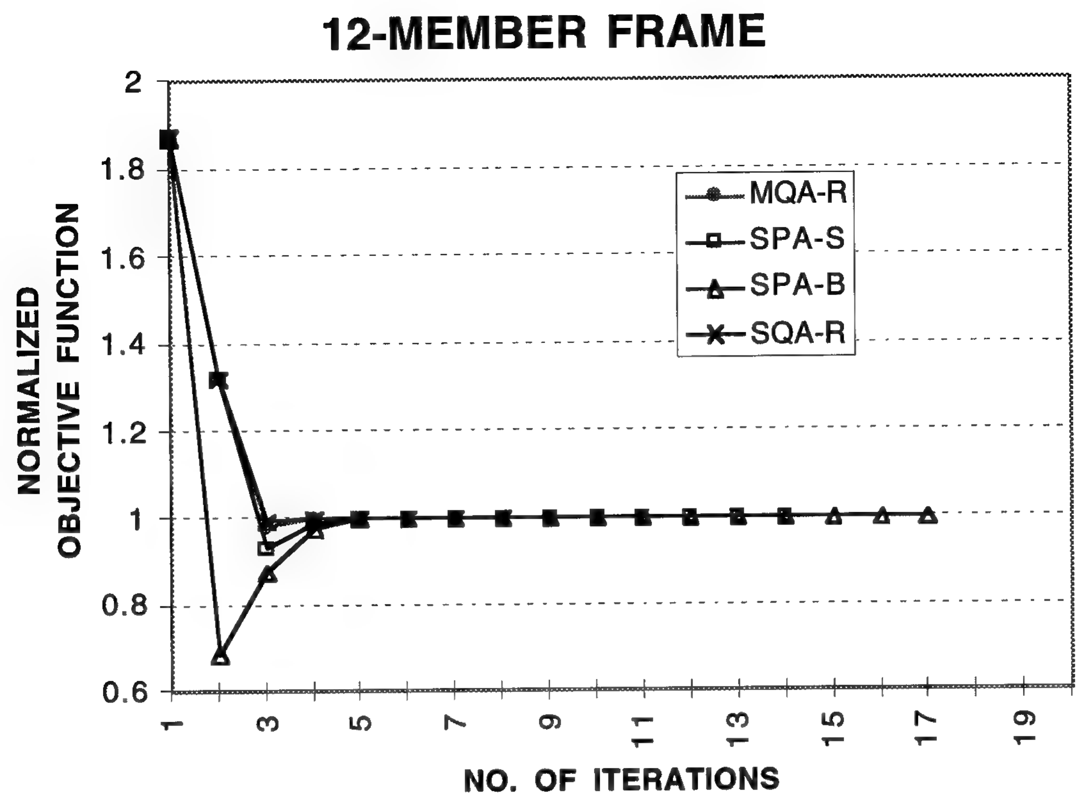
5.2.4.1.1 Direct Variable Optimization

Figure 5-11 shows the results for the three methods using direct design variables. Also shown in Table 5-17 are the exact number of iterations each method took to reach the convergence criteria. This chart also shows the CPU time each method required to converge.

Table 5-17: 12 Member Frame Convergence

Method	No. Of Iterations to Converge	Time In CPU Sec
constr	34	31
MQA-D	17	392.4
SPA-DS	46	228.4
SPA-DB	51	94.7

MQA-D did very well as compared to the other direct methods based on iterations in solving this problem; however, as shown in the table, it took a great deal more CPU time than the others. The analysis is done so quickly that the cost of calculating the approximate Hessian takes up the majority of the CPU time. This problem does demonstrate that the quadratic method can solve it in half the number of iterations; therefore, if each analysis took much longer, then a savings could be achieved. This would the case in practical design problems when the finite element model has thousands of degrees of freedom.



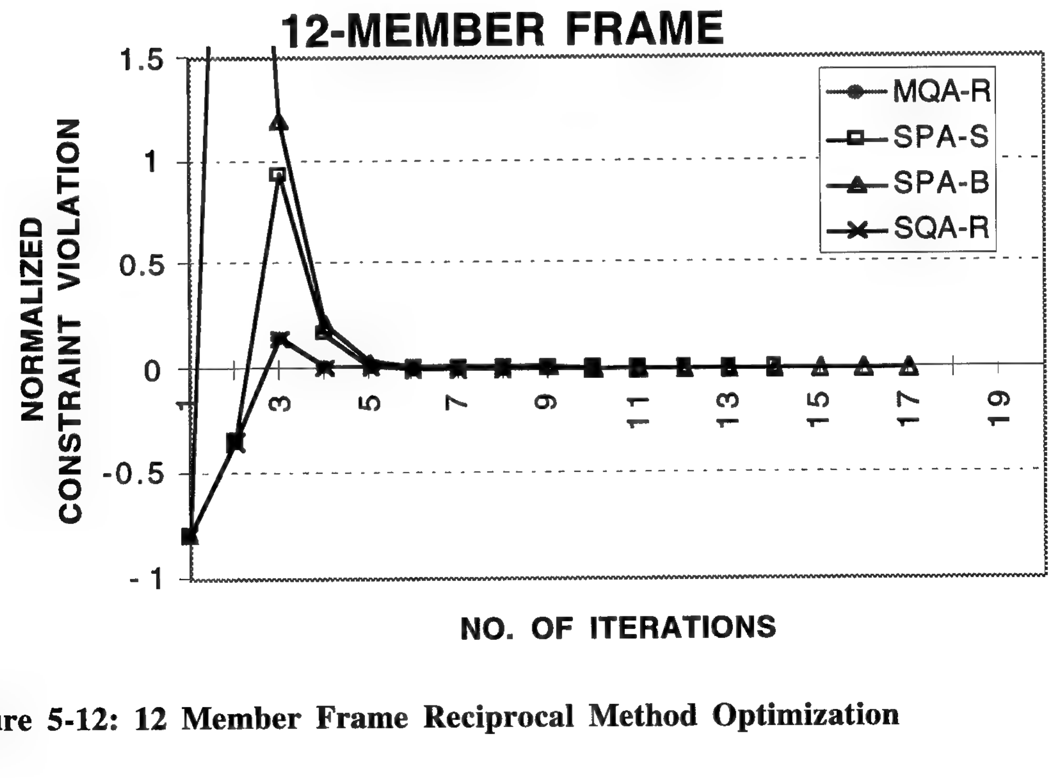


Figure 5-12: 12 Member Frame Reciprocal Method Optimization

5.2.4.1.2 Reciprocal Variable Optimization

Figure 5-12 shows the comparison of the four reciprocal methods. Table 5-18 gives the exact number of iterations to converge to the required tolerance and the CPU time.

Table 5-18: 12 Member Frame Convergence

Method	No. Of Iterations to Converge	Time In CPU Sec
MQA-R	11	114.7
SPA-RS	14	77.8
SPA-RB	17	95.4
SQA-R	8	58.0

The two quadratic methods solved this problem in fewer iterations than the single point first order reciprocal methods; however, from the graph one can see that all the methods are very close. It just takes the first order methods a little longer to hone in on the answer within tolerance. It is probably reasonable to conclude that the increased accuracy of the quadratic method speeds the final convergence.

5.2.5 ACOSS II (6 Design Variables)

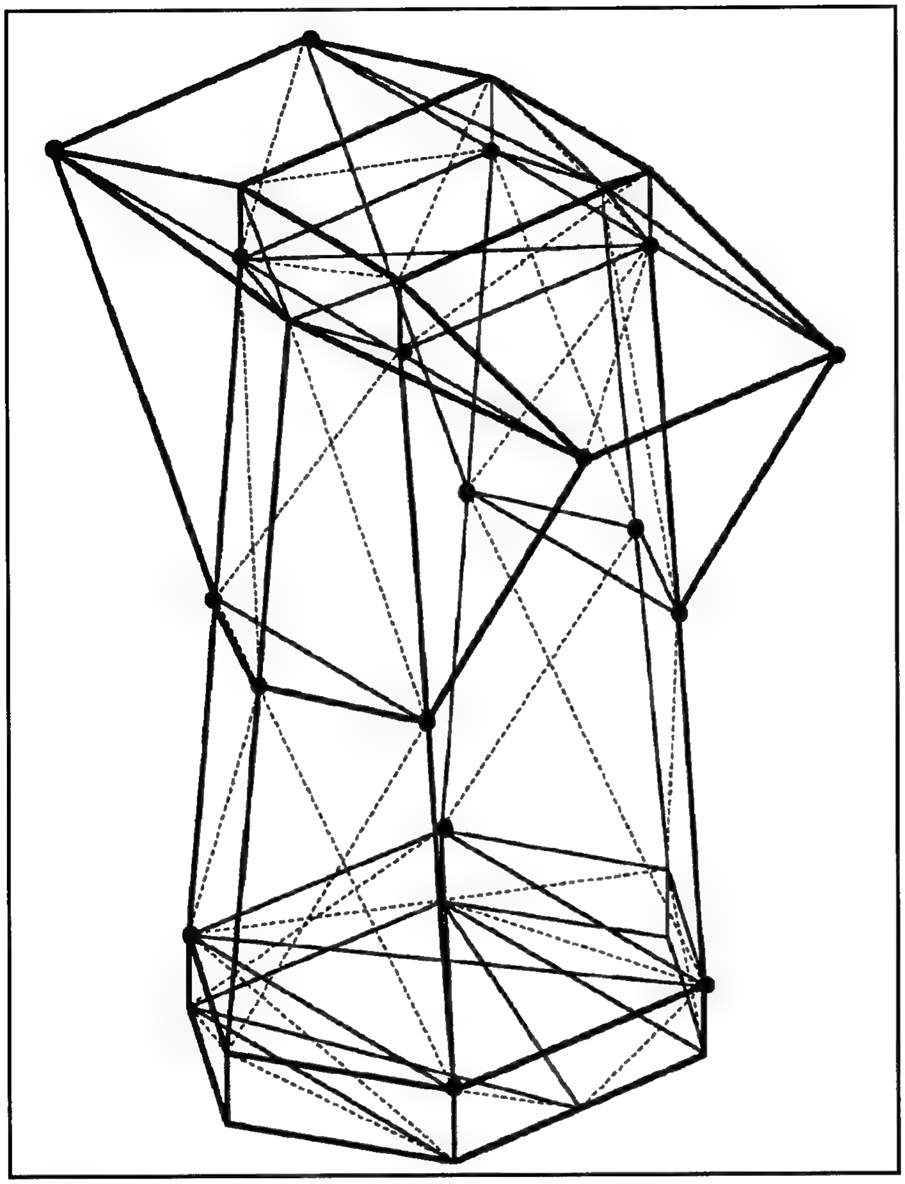


Figure 5-13: ACOSS II

5.2.5.1 Description

The Active Control of Space Structures (ACOSS) model II was developed by the Charles Stark Draper Laboratory (Canfield,1990:1121). The structure consists of two subsystems: the optical support structure and the equipment section. The two are connected by springs at three points to allow vibration isolation. In this problem the equipment section at the base was disregarded and only the optical support structure, fixed at the three connection points, was considered. The finite element model for this modified ACOSS II (with supporting cross-members added) shown in Figure 5-13 has 33 nodes, 18 concentrated masses, and 113 rod elements made of graphite epoxy with a Young's Modulus of 18.5 Mpsi, weight density of 0.055 lb/in^3 and initial areas of 10.0 in^2 for the truss members. For the purposes of this problem the structure was linked to 6 design variables subject to a lower bound frequency of 2.0 Hz, a second frequency constraint of 3.0 Hz, and minimum sizes of 0.1 in^2.

5.2.5.2 Results

5.2.5.2.1 Optimization Problem Overview

Table 5-19 summarizes the optimization problem for ACOSS II linked to six design variables. Table 5-20 shows how the 113 elements were linked. The approximate problem proved difficult to solve by "constr" in MATLAB for all the cases. Much trial and error went into finding the proper move limit so that all the cases could be compared on an equal basis.

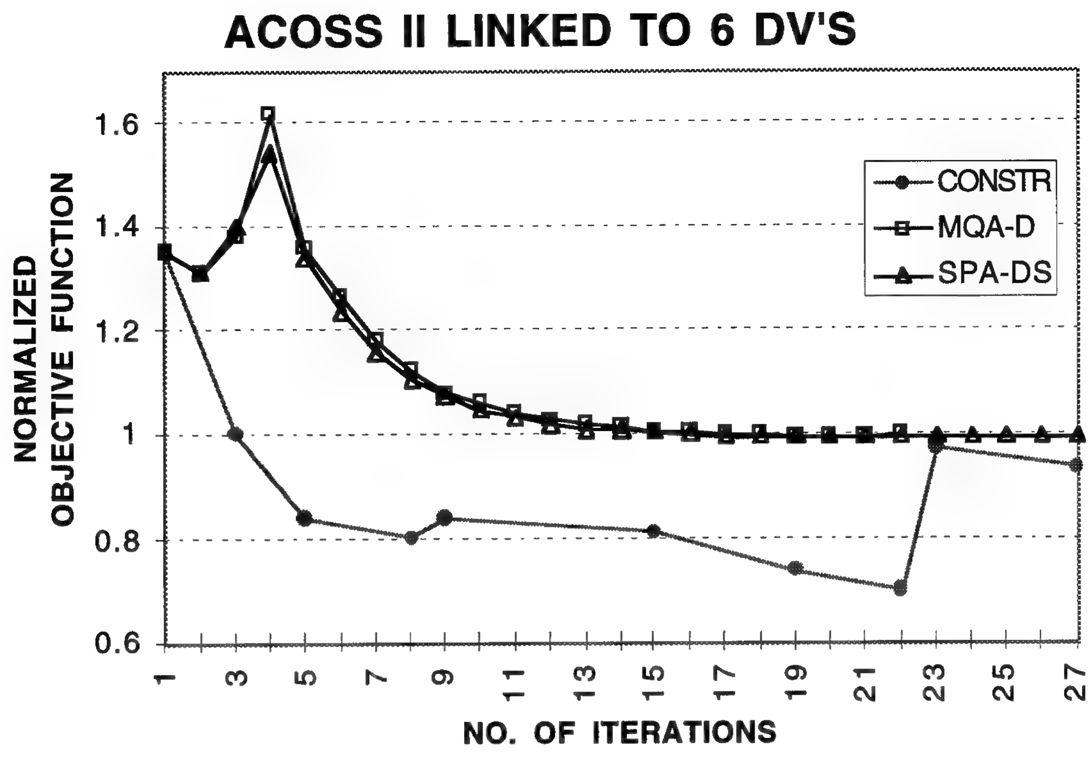
Table 5-19: ACOSS II (6) Optimization Overview

No. of Design Variables	6
No. of constraints (type)	2(Freq.)
Convergence Criteria	
Max. Change in Objective Function	5
Max. Constraint Violation	.002
Max. Change in Design Variable	.2
Move Limits	.7 with .9 reduction

Min. Allowable Design Variable	.1
Max. Allowable Design Variable	40
Initial Objective Function	17610
Initial Design Variables	10,10,10,10,10,10,
Initial Max. Constraint Violation	.637
Optimum Objective Function	13014
Optimum Design Variable Values	.22, 13.38, 33.48, 5.21, .49, 1.40

Table 5-20: ACOSS II (6) Linking

Design Variable	Elements linked to this Design Variable
1	1 2 10 11 20 23 33 34 68 69 70 71 94 98 100 104 86 88 106 107 108 109
2	4 6 44 46 48 49 52 54 21 22
3	3 9 18 19 24 25 28 29 36 37 26 27 30 35
4	31 32 12 13 14 15 16 17 45 47 50 51 53 55 56 58 60 62 64 66 57 59 61 63 65 67 72 73 76 77
5	38 39 40 41 42 43 78 80 90 92 79 93 81 82 83 84 85 87 95 97 101 103 99 105
6	74 75 89 91 96102 110 111 112 113



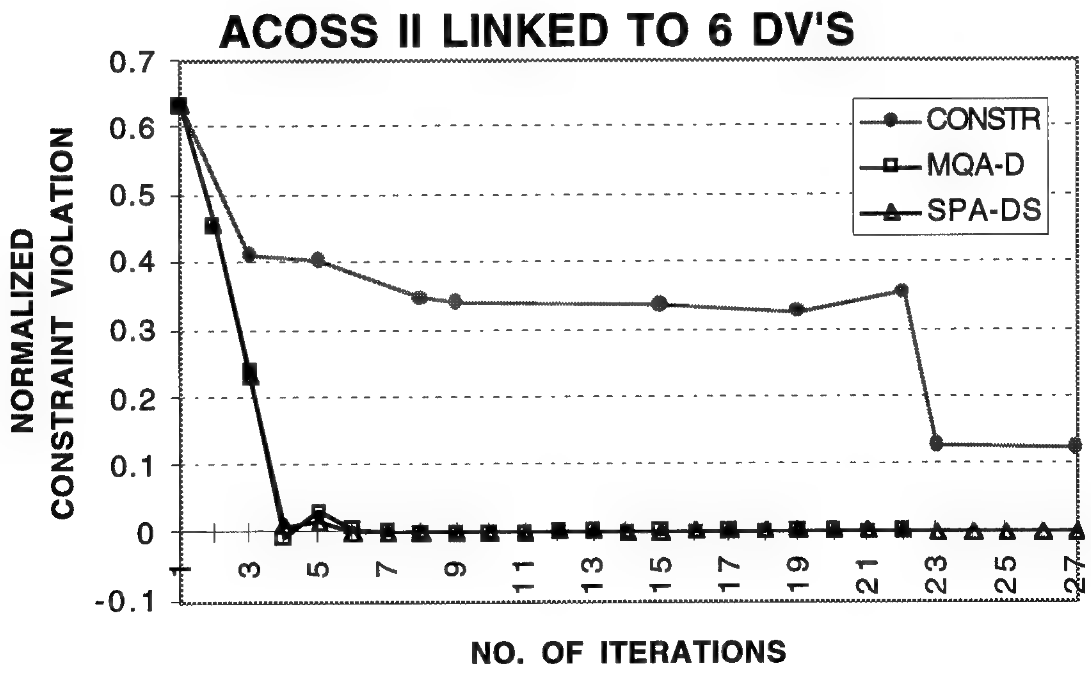


Figure 5-14: ACOSS II (6) Direct Method Optimization

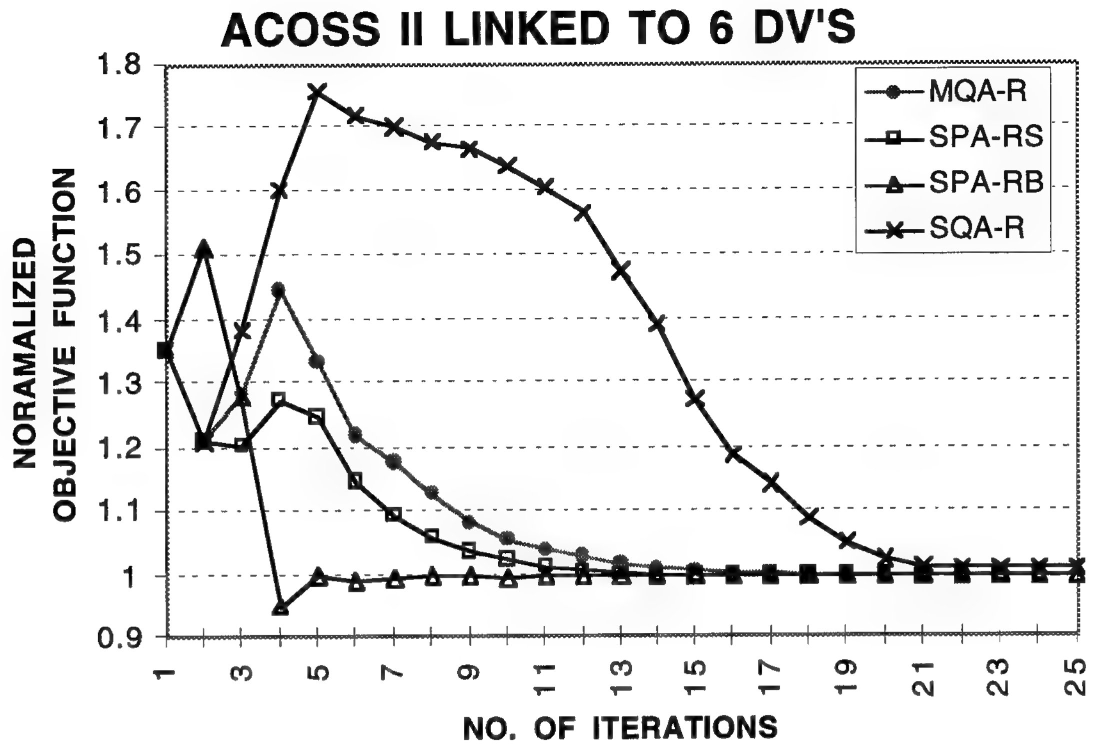
5.2.5.2.2 Direct Variable Optimization

Figure 5-14 shows the results for the three methods using direct design Table 5-21 shows the exact number of iterations each method took to reach the convergence criteria. This chart also shows the CPU time each method required to converge.

Table 5-21: ACOSS II (6) Convergence

Method	No. Of Iterations to Converge	CPU Sec
constr	42	1146.4
MQA-D	22	1036.8
SPA-DS	25	819.8

MQA-D and SPA-DS effectively solved the problem in 20 iterations, while "constr" took considerably longer to get close. For this problem the expense of the quadratic approximation paid off with MQA-D converging in only 22 iterations. SPA-DB could not provide a solution for the first iteration.



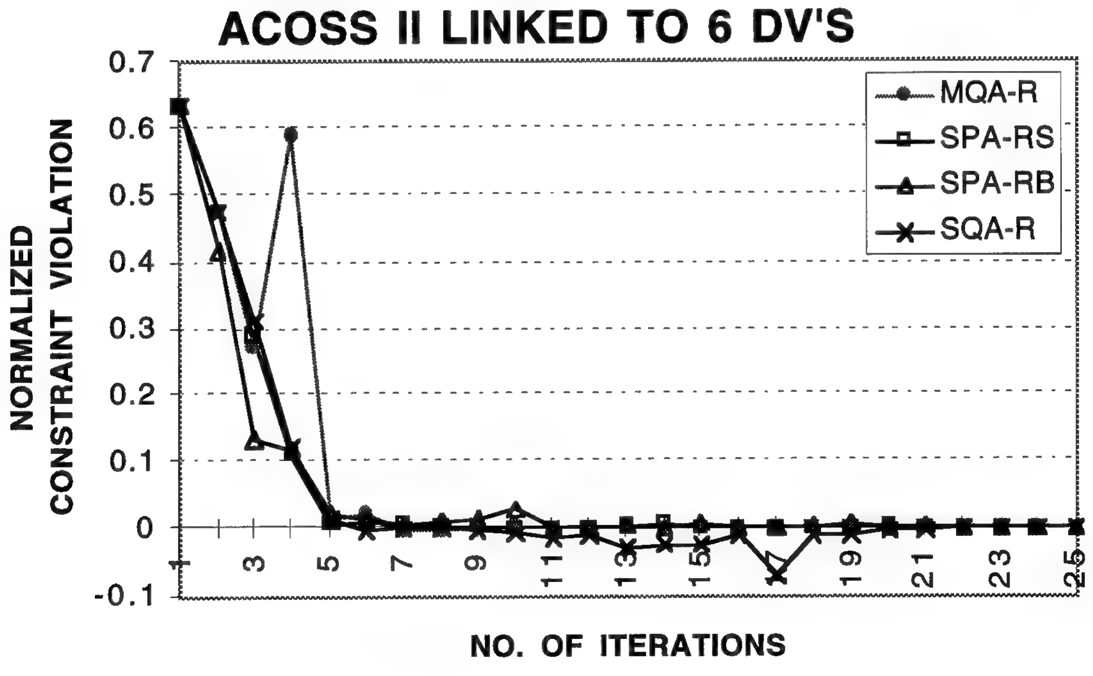


Figure 5-15: ACOSS II (6) Reciprocal Method Optimization

5.2.5.2.3 Reciprocal Variable Optimization

Figure 5-15 shows the comparison of the four reciprocal methods. Table 5-22 gives the exact number of iterations and the CPU time to converge to the required tolerance.

Table 5-22: ACOSS II (6) Convergence

Method	No. Of Iterations to Converge	CPU Sec
MQA-R	19	1068.9
SPA-RS	19	864.3
SPA-RB	20	938.3
SQA-R	25+	908.8

The reciprocal variables did just slightly better than the direct for this problem. It was interesting that this is first case where both of the quadratics did not show any improvement over the first order approximations. SQA-R had some difficulty with this problem, stopping at 25 iterations, about 150 lb. above the optimum.

5.2.6 ACOSS II (31 Design Variables)

5.2.6.1 Description

The ACOSS II model here is exactly the same as the one above except that the structure is linked to 31 design variables. Table 5-24 shows how the 113 elements are linked to 31.

5.2.6.2 Results

5.2.6.2.1 Optimization Problem Overview

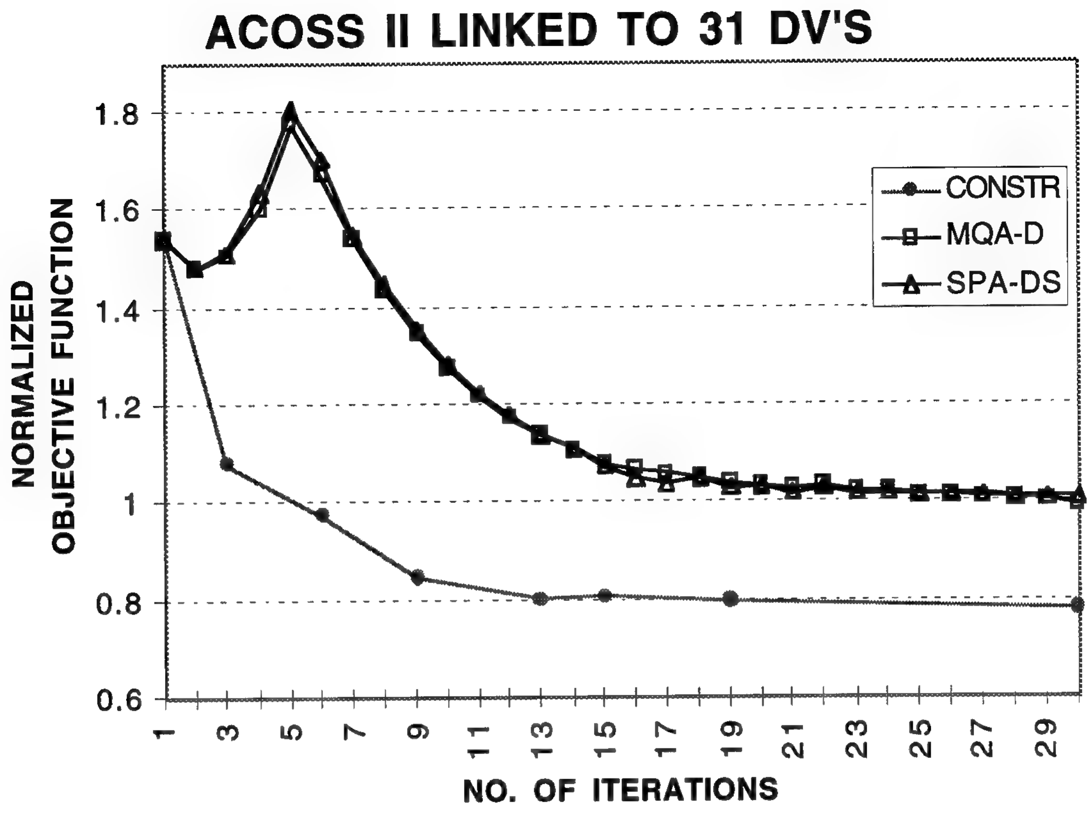
Table 5-23 gives an overview of the optimization problem for ACOSS II linked to 31 Design variables.

Table 5-23: ACOSS II (31) Optimization Overview

No. of Design Variables	31
No. of constraints (type)	2(Freq.)
Convergence Criteria	
Max. Change in Objective Function	5
Max. Constraint Violation	.002
Max. Change in Design Variable	.2
Move Limits	.8 with .98 reduction except for MQA-R(1,.98)
Min. Allowable Design Variable	.1
Max. Allowable Design Variable	40
Initial Objective Function	17610
Initial Design Variables	10 for all 31
Initial Max. Constraint Violation	.637
Optimum Objective Function	11705
Optimum Design Variable Values	.10,12.65,21.08,34.53,.10,35.14,7.29, 4.93,12.37,3.09,.70,10.98,27.47,32.18, .10,4.09,5.19,.10,2.84,1.78,.25,.1,1.21, .28,.5836,.10,.56,.45,.10,.95

Table 5-24: ACOSS II (31) Linking

Design Variable	Elements linked to this Design Variable
1	1 2 10 11
2	4 6
3	3 9
4	24 25 18 19
5	20 23
6	28 29 36 37
7	31 32
8	12 13 14 15 16 17
9	44 46 48 49 52 54
10	45 50 47 51 55 53
11	38 39 40 41 42 43
12	21 22
13	26 27
14	30 35
15	33 34
16	56 58 60 62 64 66
17	57 59 61 63 65 67
18	68 69 70 71
19	72 73 76 77
20	74 75 89 91
21	78 80 90 92
22	98 94 100 104
23	96 102
24	79 93
25	81 82 83 84
26	86 88
27	85 87
28	95 97 101 103
29	99 105
30	106 107 108 109
31	110 111 112



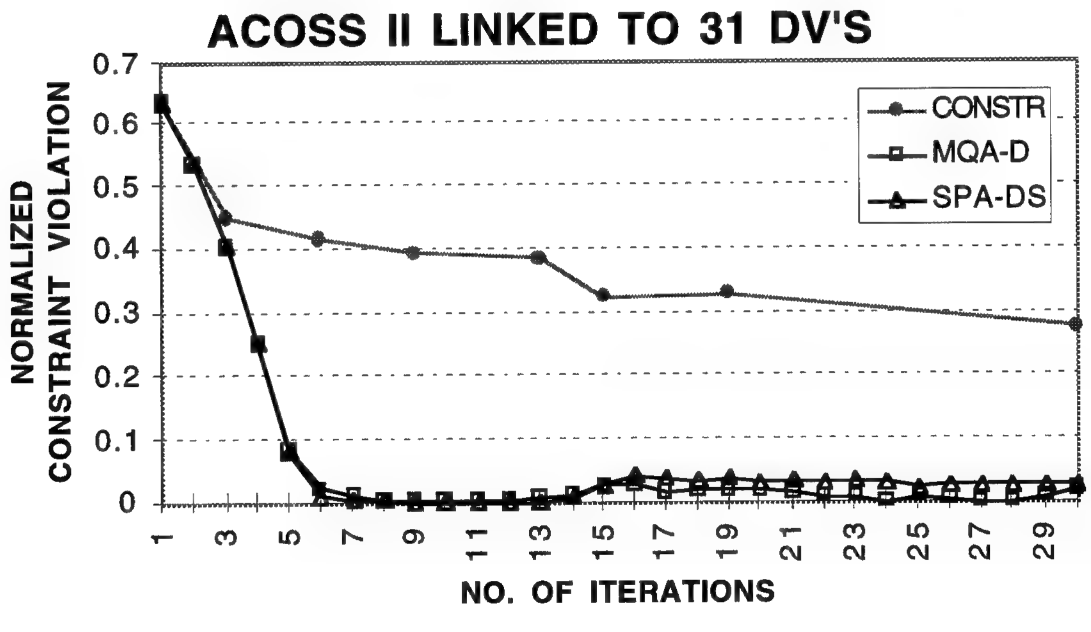


Figure 5-16: ACOSS II (31) Direct Method Optimization

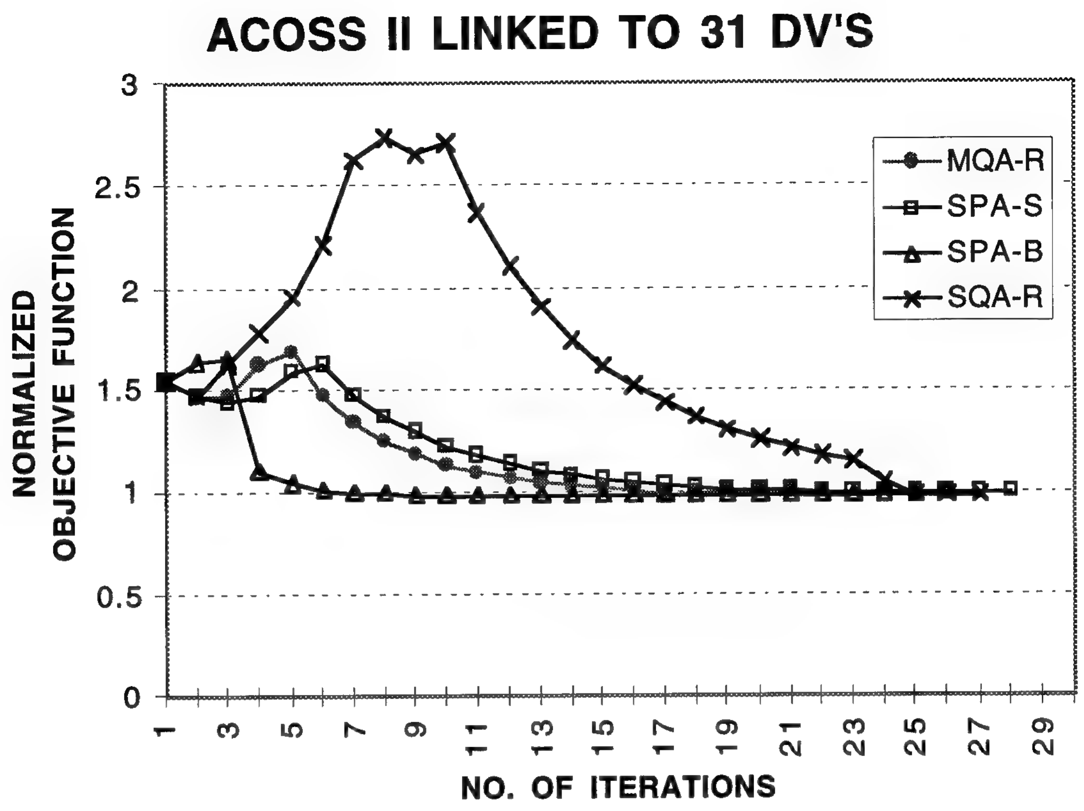
5.2.6.2.2 Direct Variable Optimization

Figure 5-16 shows the results for the three methods using direct design variables. Also shown in Table 5-25 are the exact number of iterations and the CPU time each method took to reach the convergence criteria.

Table 5-25: ACOSS II (31) Convergence

Method	No. Of Iterations to Converge	CPU Sec
constr	200	5712
MQA-D	36	14896
SPA-DS	49+	3919.2

As with ACOSS II linked to six variables this problem proved difficult to solve and several trial and error cases were needed to find the proper move limit so that the optimizer in MATLAB could solve the approximate problem. As with the six variable case SPA-DB would get stuck on the second iteration. MQA-D has a clear advantage in solving this problem when only the number of iterations is considered. It gives useful results in about 20 iterations. However, to let it proceed past this point cost a substantial amount of CPU time. The large amount of CPU time required to solve the problem in the quadratic case is related to the fact that the finite element analysis has to solve a problem with approximately 90 degrees of freedom; whereas, the pseudo-inverse calculations involve finding 496 unknowns in the Hessian. The Hessian calculation is expected to be much more expensive. However, if the problem had 31 design variables and many thousands of degrees of freedom then estimating the Hessian would not have used the majority of the CPU time and MQA-D would have been more economical.



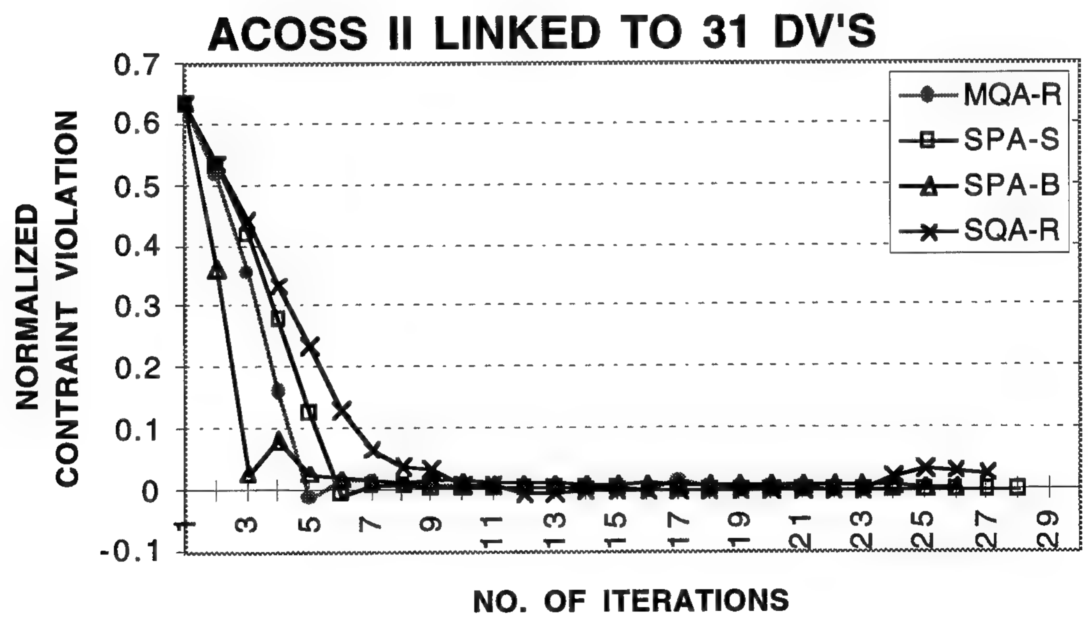


Figure 5-17: ACOSS II (31) Reciprocal Optimization

5.2.6.2.3 Reciprocal Variable Optimization

Figure 5-17 shows the comparison of the four reciprocal methods. Table 5-26 gives the exact number of iterations and the CPU time to converge to the required tolerance.

Table 5-26: ACOSS II (31) Convergence

Method	No. Of Iterations to Converge	CPU Sec
MQA-R	26*	8464
SPA-RS	28	2590
SPA-RB	86	N/A
SQA-R	52+	4635

* Used different move limits

MQA-R solved this problem in the least iterations but it required a different move limit than the rest for MATLAB to solve the approximate problem. The move limit began at 1 and was reduced by .98 at each iteration. This move limit would not work for the others. SPA-RB came close to the optimum in about 15 iterations but it could not converge until 86 iterations. It just bounced around the optimum until the move limit was reduced small enough to force it to converge. SQA-R took a very long and conservative route to the optimum. It was close in about 25 iterations but still did not meet the convergence tolerance in 52 iterations. Since the move limits are different it is not really fair to compare, but MQA-R does solve this problem in fewer iterations, even though it used a great deal of CPU time. However, the two single point reciprocal methods are very close for a useable solution.

5.2.7 40 Member Frame

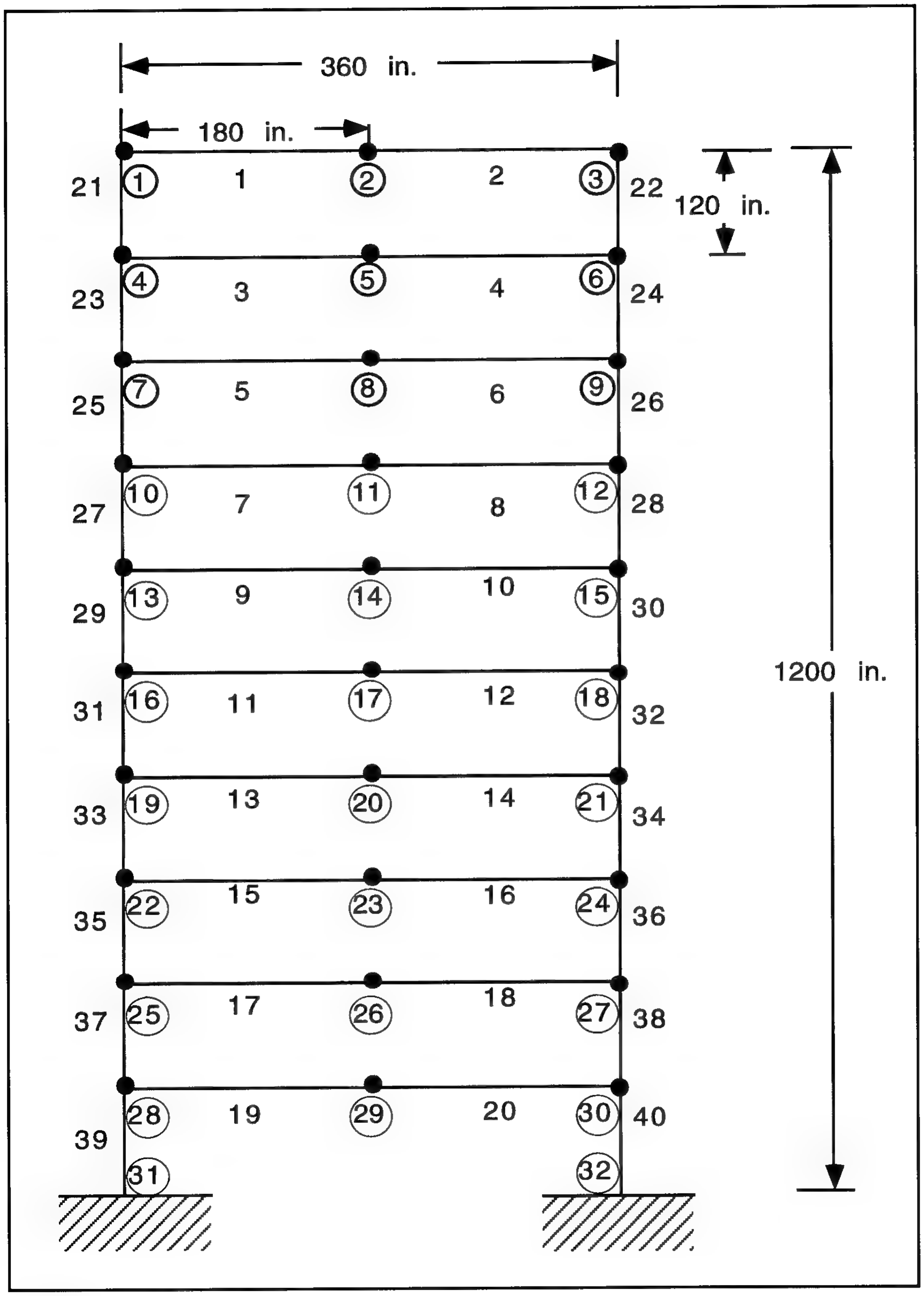


Figure 5-18: 40-Member Frame

5.2.7.1 Description

The 40-member plane frame shown in Figure 5-18 is made up of I-section members with 32 nodes with nodes 31 and 32 restrained from displacement. It was subjected to 3 load cases with 288 displacement constraints and 120 stress constraints shown in Table 5-27. For this structure there is a fixed relationship between the moment of inertia(I), section modulus (S) and the cross sectional area (A). $I=0.20720xA^3$, $S=0.39300xA^2$ (Kolonay,1987:109-115), where the constant, 0.20720, has units of in^{-2} and the constant, 0.39300, has units of in^{-1}. Material and element properties are given in Table 5-28 and Table 5-29.

Table 5-27: 40-Member Frame Load Cases

		Direction of load	
Load Case	Node	x(lb)	y(lb)
1	1		-6000
	2		-24000
	3		-6000
	4		-12000
	5		-48000
	6		-12000
	7		-12000
	8		-48000
	9		-12000
	10		-12000
	11		-48000
	12		-12000
	13		-12000
	14		-48000
	15		-12000
	16		-12000
	17		-48000
	18		-12000
	19		-12000
	20		-48000
	21		-12000
	22		-12000
	23		-48000
	24		-12000
	25		-12000

	Direction of load		
Load Case	Node	x(lb)	y(lb)
	26		-48000
	27		-12000
	28		-12000
	29		-48000
	30		-12000
2	1	3180	-4500
	2		-18000
	3		-4500
	4	6230	-9000
	5		-36000
	6		-9000
	7	6080	-9000
	8		-36000
	9		-9000
	10	5920	-9000
	11		-36000
	12		-9000
	13	5740	-9000
	14		-36000
	15		-9000
	16	5540	-9000
	17		-36000
	18		-9000
	19	5300	-9000
	20		-36000
	21		-9000
	22	5000	-9000
	23		-36000
	24		-9000
	25	4610	-9000
	26		-36000
	27		-9000
	28	4010	-9000
	29		-36000
	30		-9000
3	1		-4500
	2		-18000
	3	-3180	-4500
	4		-9000
	5		-36000
	6	-6230	-9000
	7		-9000
	8		-36000
	9	-6080	-9000
	10		-9000

	Direction of load		
Load Case	Node	x(lb)	y(lb)
	11		-36000
	12	-5920	-9000
	13		-9000
	14		-36000
	15	-5740	-9000
	16		-9000
	17		-36000
	18	-5540	-9000
	19		-9000
	20		-36000
	21	-5300	-9000
	22		-9000
	23		-36000
	24	-5000	-9000
	25		-9000
	26		-36000
	27	-4610	-9000
	28		-9000
	29		-36000
	30	-4010	-9000

Table 5-28: 40-Member Frame Material Properties

Material	Steel
Modulus of Elasticity	$29x10^6$ psi
Specific Weight	.283 lb/in^2
Upper Limit on Area	88.0 in^2
Lower Limit on Area	2 in^2
Number of Loading Conditions	3
Displacement limits(All)	x(2 in^2), y(10 in^2),z(50.rad)

Table 5-29: 40-Member Frame Element Properties

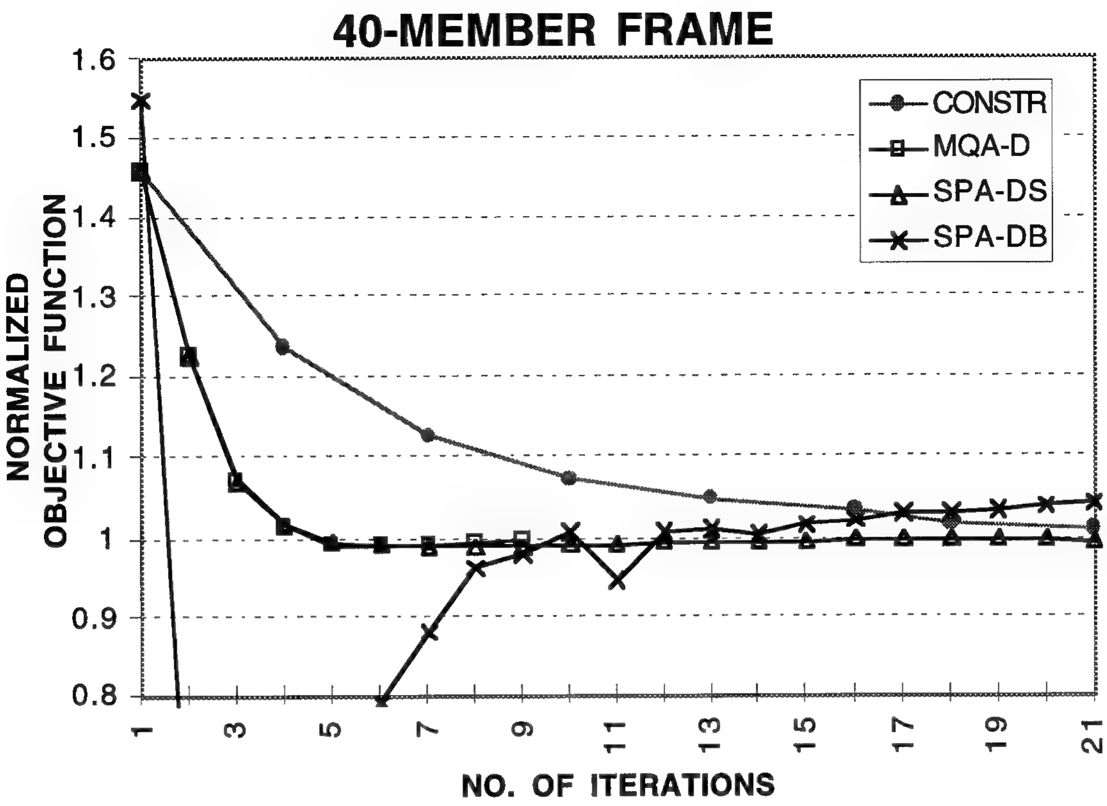
Area	30 in^2
Section Modulus	353.7 in^3
Moment of Inertia	5594 in^4
Maximum Allowable Stress	22000 psi
Initial Weight	50940 lb

5.2.7.2 Results

Table 5-30 gives an overview of the information used in this optimization problem. All of the methods converged to the same optimum.

Table 5-30: 40-Member Frame Optimization Overview

No. of Design Variables	40
No. of constraints (type)	288(Disp), 120(Stress)
Convergence Criteria	
Max. Change in Objective Function	10
Max. Constraint Violation	.002
Max. Change in Design Variable	.5
Move Limits	1 with .9 reduction
Min. Allowable Design Variable	.5
Max. Allowable Design Variable	88
Initial Objective Function	50940
Initial Design Variables	30 for all
Initial Max. Constraint Violation	-.0969
Optimum Objective Function	34894
Optimum Design Variable Values	12.41,12.41,16.97,16.97,19.60,19.66, 21.72,21.70,23.21,23.24,24.40,24.40, 25.29,25.38,26.33,26.31,26.30,26.28, 15.89,15.89,10.60,10.60,12.88,12.92, 14.75,14.65,16.02,16.00,17.13,17.22, 18.85,18.83,20.54,20.58,22.32,22.30, 24.64,24.60,37.86,37.66



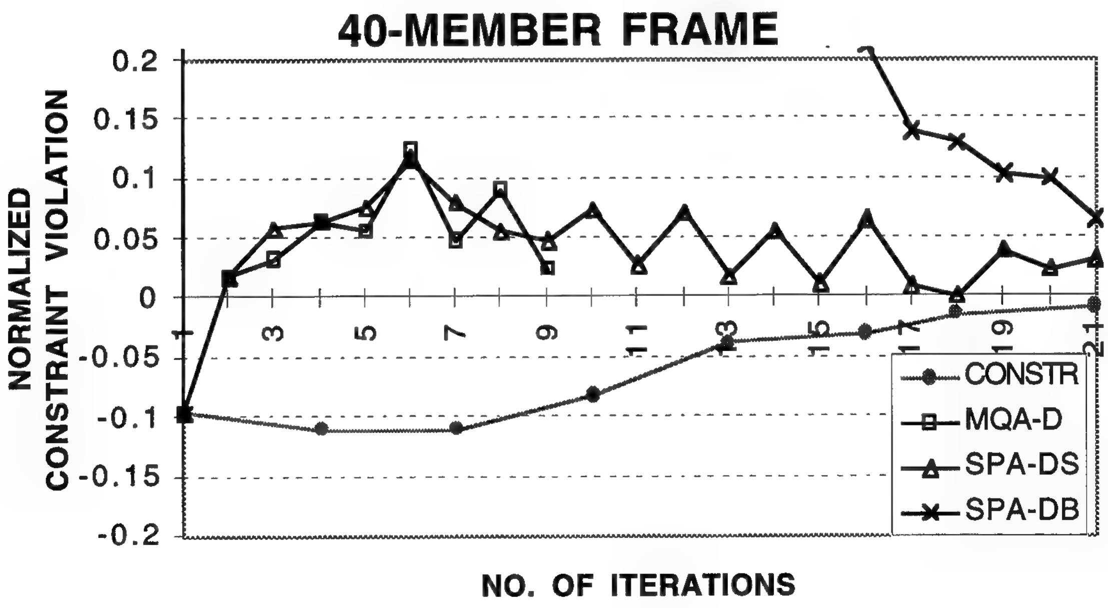


Figure 5-19: 40-Member Frame Direct Optimization

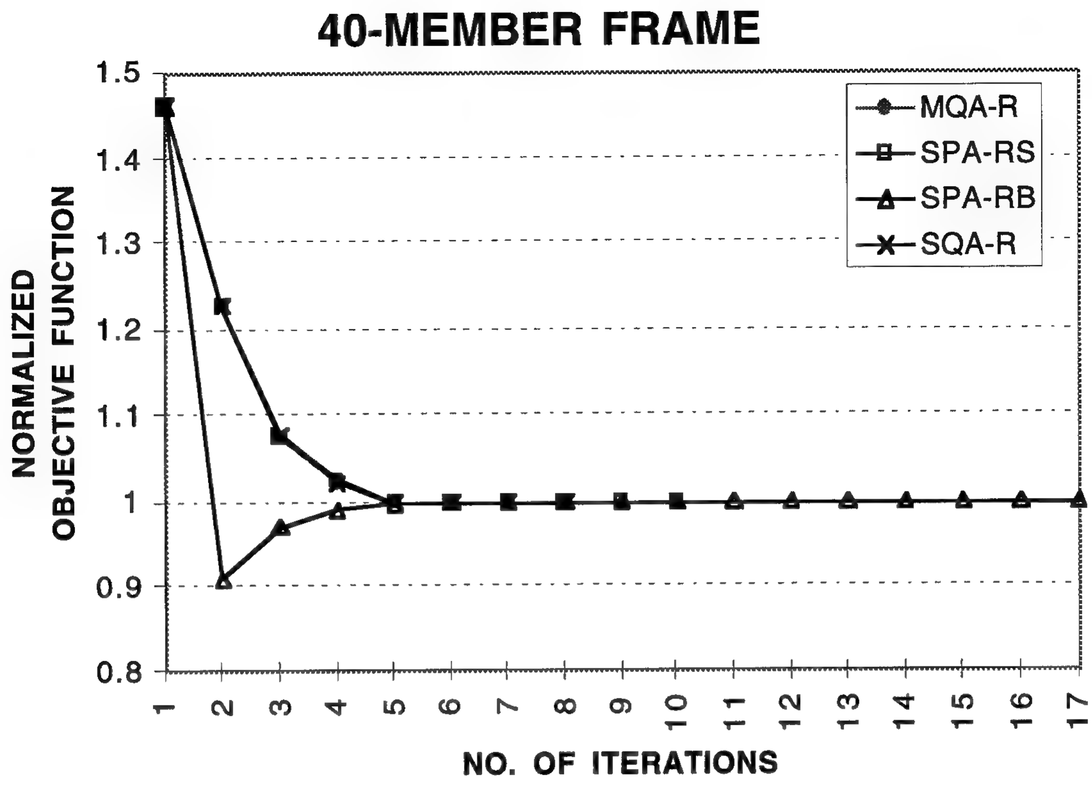
5.2.7.2.1 Direct Variable Optimization

Figure 5-19 shows the results for the three methods using direct design variables. Table 5-31 shows the exact number of iterations each method took to reach the convergence criteria. This table also shows the CPU time each method required to converge.

Table 5-31: 40-Member Frame Convergence

Method	No. Of Iterations to Converge	Time In CPU Sec
constr	42	428
MQA-D	9+	16378.5
SPA-DS	48	8309.7
SPA-DB	43	1149.9

MQA-D and SPA-DS were able to reduce the objective function to near optimum within about five iterations, but they both had trouble meeting the constraint tolerance. MQA-D was only able to run nine iterations because of the time each iteration was taking. Table 5-31 shows the amount of CPU time required was much greater than the others. Also, the time was increasing greatly for each iteration. This problem demonstrates a weakness of this method. The amount of time to calculate the approximate Hessian for this method has become prohibitively expensive as the number of design variables and previous points increases.



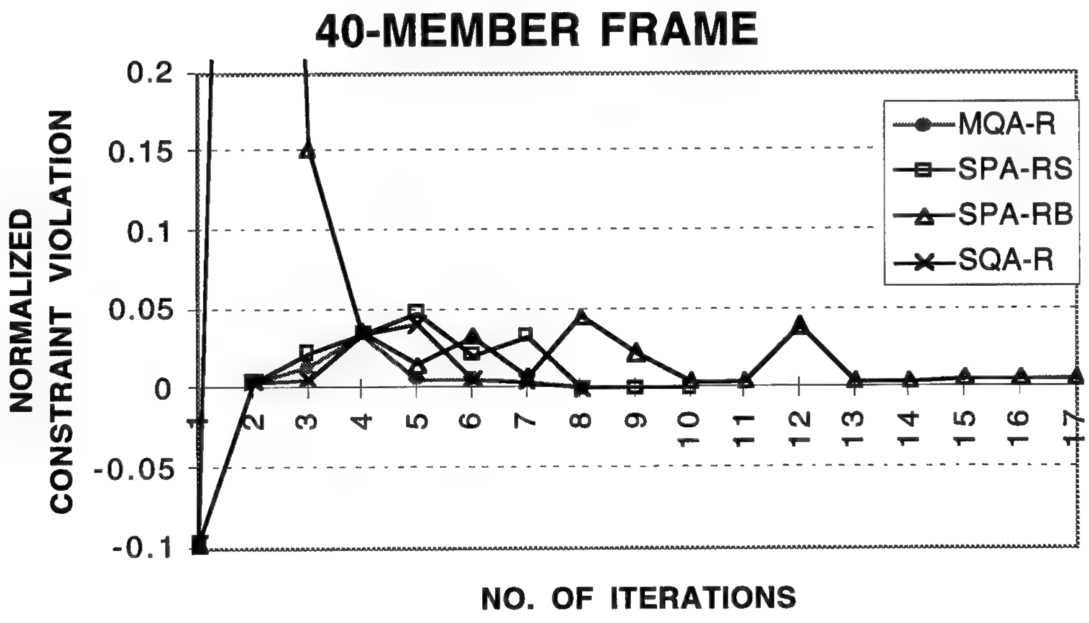


Figure 5-20: 40-Member Frame Reciprocal Optimization

5.2.7.2.2 Reciprocal Variable Optimization

Figure 5-20 shows the comparison of the four reciprocal methods. Table 5-32 gives the exact number of iterations to converge to the required tolerance and the CPU time.

Table 5-32: 40-Member Frame Convergence

Method	No. Of Iterations to Converge	Time In CPU Sec
MQA-R	8+	11755
SPA-RS	10	742.6
SPA-RB	26+	1413.4
SQA-R	8	903.3

As with MQA-D above, MQA-R has become prohibitively expensive for this problem. It is doing a very good job for each iteration. In eight iterations it has almost converged but each iteration takes an unreasonable amount of time relative to the expense of the exact analysis. This is expected, since the finite element analysis is solving for 90 degrees of freedom versus a pseudo-inverse calculation for the 820 unknown terms in the approximate Hessian. SPA-RS and SQA-R both do just as well or better using much less CPU time. The reciprocal methods for this problem converge so quickly that very little Hessian information is built up for the few previous points as compared to the number of design variables. This problem confirms the presumption that MQA is probably only useful for problems with few design variables and constraints.

6. Conclusion

6.1 Overview

Table 6-1 is a summary of the number of iterations needed to meet the convergence criteria. Several comparisons can be made from this table. It should be emphasized that the purpose of the comparison is to look at general trends in solving the different structures. Other structures may or may not have similar results, but I believe that these results will reflect reasonable expectations for solving similar type problems. Several comparisons can be made, reciprocal vs. direct variables, quadratic vs. linear approaches, multipoint vs. two point quadratic approximations, and spherical vs. box move limits.

Table 6-1: Optimization Summary

	Direct Variable Methods				Reciprocal Variable Methods			
	constr	MQA-D	SPA-DS	SPA-DB	MQA-R	SPA-RS	SPA-RB	SQA
5-Section Beam	16	11	20	14	6	16	12	6
4-Member Frame	47	38	116+	40	5	14	12	12
10 Bar Truss	34	35	50+	73+*	14	14	14	36+
12 Member Frame	34	17	46	51	11	14	17	8
ACOSS II (6)	42	22	25	N/A	19	19	20	25+
ACOSS II (31)	200	36	49+	N/A	26*	28	86	52+
40 Member Frame	42	9+	48	43	8+	10	26+	8

* Different Move Limits than others for same structure

6.2 Reciprocal vs. Direct

As expected, reciprocal variable approximations performed much better in almost every case. This is typical with structural problems with stress and displacement constraints. With the two ACOSS models the difference was not as great, probably because the frequency constraints tend to be highly non-linear and difficult to approximate.

6.3 Quadratic vs. Linear

The focus of this thesis was to test a quadratic approximation, specifically MQA. For the direct variable cases MQA-D converged in the fewest number of iterations in six out of seven cases. For the 12 member frame it converged in less than half the number of iterations of the other direct methods. MQA-D does appear to have a drawback, shown from the cases where CPU times were compared. It is computationally expensive. Calculating the approximate Hessian involves a pseudo-inverse which solves for n(n+1)/2 elements in the Hessian matrix. Where n is the number of design variables. Unless the finite element analysis is of this order or larger, then it is doubtful that any benefit will be obtained with regard to the overall computational expense of solving the optimization problem. The 40 Member Frame and ACOSS linked to 31 design variables demonstrate this because for these problems the Hessian is much more expensive to calculate than the finite element analysis. However, if the number of design variables and constraints are not too large MQA could definitely prove worthwhile when function evaluations are very expensive, since MQA for direct variables does show promise in solving problems in fewer iterations. For the reciprocal variables MQA still did very well, but as it was already known that reciprocal variables generally do a very good job in approximating structural problems, very little was gained in most cases by using the quadratic approximation. It did help with accuracy when zeroing in on convergence criteria. The same drawback of high computational expense for large numbers of variables found in the MQA results in direct variables, applies to MQA for reciprocal variables.

6.4 Sequential Two Point Quadratic vs. Multipoint Quadratic

Of the cases tested, the multipoint was quicker to converge in three, slower in three, and the same in one as compared to the sequential. From this data, the only conclusion is that using multiple points probably does not increase the accuracy of the approximation greatly over using sequential information updated based on the two most recent points. However, overall the multipoint approximation was much more stable in converging than the sequential. SQA-R converged slowly for three of the seven problems.

6.5 Spherical Vs. Box Move Limits

It is not completely fair to compare the two move limit types without some criteria to equate an equal spherical move limit to some box limit. No good way to exactly compare these has yet been developed, but from the data obtained here, it appears that whether or not one type is better than the other is very case dependent. The two ACOSS problems would not run properly with box move limits and the 10 bar truss had to be solved with smaller move limits using box limits, because with the same move limit as the other direct methods it would diverge. Generally, it appears that the spherical move limits result in a more stable if not faster convergence for most of the problems as shown by the graphs in chapter 5.

6.6 Recommendations

The next step in testing MQA is to not use every previous point but to pick a group of the most recent or closest points to see if this will increase the computationally efficiency without sacrificing accuracy. Also, if SQA strengths and weaknesses can be predicted it may be very useful to solve certain types of problems or perhaps it can be modified to perform well in more cases.

6.7 The Bottom Line

MQA does do a good job of approximating the functions to be used in optimization. How much better is dependent on the particular problem. In all cases, its convergence was very stable and it would seem to be very reliable. However, whether or not the computational expense of calculating the second order approximation using MQA is computationally efficient will depend on the relative size of the analysis and the optimization problems. MQA is recommended only for problems with few design variables and very expensive analysis of functions (more expensive than a singular value decomposition, which is the majority of the expense in calculating the pseudo-inverse). Also, it is recommended for problems where appropriate intermediate variables, such as reciprocal variables for structural problems, are not known.

Appendix A: Multipoint Quadratic Approximation Matlab Routine

```
function[x,OPTIONS]=mqa(FUN,X0,OPTIONS,VLB,VUB,GRADFUN,APX_TYPE,options)
% MQA Finds the constrained minimum of a function of several variables
%           by Multipoint Quadratic Approximations.
%
%     X=MQA('FUN',X0) starts at X0 and finds a constrained minimum to
%     the function which is described in FUN (usually an M-file: FUN.M).
%     The function 'FUN' should return two arguments: a scalar value of the
%     function to be minimized, F, and a matrix of constraints, G:
%     [F,G]=FUN(X). F is minimized such that G < zeros(G).
%
%     X=MQA('FUN',X,OPTIONS) allows a vector of optional parameters to
%     be defined. For more information type HELP FOPTIONS.
%     OPTIONS(15) = move limit reduction factor >1 (default=1)
%   OPTIONS(18) = move limit ratio (default=0.5 = 50%)
%
%     X=MQA('FUN',X,OPTIONS,VLB,VUB) defines a set of lower and upper
%     bounds on the design variables, X, so that the solution is always in
%     the range VLB < X < VUB.
%
%     X=MQA('FUN',X,OPTIONS,VLB,VUB,'GRADFUN') allows a function
%     'GRADFUN' to be entered which returns the partial derivatives of the
%     function and the  constraints at X:  [gf,GC] = GRADFUN(X).

%     X=MQA('FUN',X,OPTIONS,VLB,VUB,'GRADFUN',APX_TYPE)
%     specifies approximation type for each function (obj & constraints) where
%     1 = 1st order
%     2 = Quadratic
%     3 = 1st order Reciprocal
%     4 = Quadratic Reciprocal
alpha=1
% Written by Michael A.Blaylock & Robert A. Canfield

% Process input arguments.
fcnstr = [FUN, '(x)'];
ndv = length(X0);
defaultOPTIONS = [1 0.001 0.001 1.e-4];
if nargin < 3,
      OPTIONS=defaultOPTIONS;
else
      m = length(OPTIONS);
      for n=length(defaultOPTIONS):-1:2,
            if m<n, OPTIONS(n)=0; end;
            if OPTIONS(n)==0, OPTIONS(n) = defaultOPTIONS(n); end;
      end
      if m<1, OPTIONS(1) = 1; end
end;
OPTIONS = foptions(OPTIONS);
if OPTIONS(14)==0, OPTIONS(14)=ndv*10; end
if OPTIONS(15)==0, OPTIONS(15)=1; end
if OPTIONS(18)==0, OPTIONS(18)=0.5; end
if nargin < 4, VLB=[]; end
if nargin < 5, VUB=[]; end
if nargin < 6, GRADFUN=[]; end
if length(GRADFUN)
        grdstr = [GRADFUN, '(x)'];
else
        error('Finite Difference Gradient not yet implemented')
end
if nargin < 7, APX_TYPE=[]; end

% Set options for approximate problem relative to exact, unless given.
if nargin < 8,
        options = OPTIONS;
        options(1) = 0;
        options(4) = OPTIONS(4)/10;
else
        m = find(options==0);
        options(m) = OPTIONS(m);
        if length(options)<4 | any(m==4), options(4) = OPTIONS(4)/10; end;
end
if OPTIONS(1)==2, options(1) = 1; end;

% Error check the lower/upper bounds and initial guess.
lenvlb=length(VLB); if lenvlb, xlb = VLB(:); end;
lenvub=length(VUB); if lenvub, xub = VUB(:); end;
xlb(lenvlb+1:ndv,1) = -Inf*ones(ndv-lenvlb,1);
xub(lenvub+1:ndv,1) =  Inf*ones(ndv-lenvub,1);
if lenvlb*lenvub>0
        range = 1:min(lenvlb,lenvub);
        if any(VLB(range)>VUB(range)), error('Bounds Infeasible'), end
end
x  = X0(:);
dx = zeros(size(x));
if any(x<xlb) | any(x>xub)
        disp('Initial X vector--not within bounds--has been reset')
        x = max( x, xlb );
        x = min( x, xub );
end
violated = abs(OPTIONS(4));
active = -10*violated;
movred = abs(OPTIONS(15));
movmax = abs(OPTIONS(18));
movmin = 0;
move   = movmax;
if movred < 1, movred = 1/movred; end;

% Initial function evaluations.
[f, g(:)] = eval(fcnstr);
[mg,mj]   = max(g);
ncon      = length(g);
if length(g) < 1, error('Constraints must be defined'), end
gapx=NaN*ones(size(g));
bound = ' ';
HessF = zeros(ndv,ndv);
```

```
apx_order = 2*ones(1,ncon+1);
apx_order(1:length(APX_TYPE)) = APX_TYPE(:)';
apx_f = apx_order(1);
apx_g = apx_order(2:ncon+1);
% First order for first iter.
m = find( apx_g==2 | apx_g==4 );
apx_g(m) = apx_g(m)-1;

% Set up iteration count and optional print.
niter    = 0;
countfcn = 1;
countgrd = 0;
if OPTIONS(1)>0
      disp('')
      disp('f-COUNT   FUNCTION        MAX{g}  j    gapx_j   Move_limit
max_dx');
end

%--------------------------------Main Loop-----------------------------
converged = 0;
while converged ~= 1
      niter = niter + 1;
      fold  = f;
      x0    = x;
      feasible_start = mg < violated;
      if OPTIONS(1)>0
          [m,n] = max( abs(dx) );
            disp([sprintf('%5.0f %12.6g %12.6g %3.0f %9.3g %9.3g %9.3g',...
            countfcn,f,mg,mj,gapx(mj),move,dx(n)) bound]);
      end

%   Solve quadratic problem.
      [gradf, gradg] = eval(grdstr);
      xstore = [xstore,x(:)];
      gstore = [gstore,g(:)];
      dgstore = [dgstore,gradg];
      if apx_f==2
            fstore = [fstore,f];
            dfstore = [dfstore,gradf];
            HessF = hapx( fstore, dfstore, xstore );
            if OPTIONS(1)==3, HessF, end
      elseif apx_f==4
            fstore = [fstore,f];
            dfstore = [dfstore,gradf.*(-x0.^2)];
            HessF = hapx( fstore, dfstore, 1./xstore );
      end
      for j=1:ncon
      dg=[];
            if apx_g(j)==2
                  Hess = hapx( gstore(j,:)', dgstore(:,j:ncon:j+(niter-
1)*ncon), xstore,alpha );
                  HessG(:,j) = Hess(:);
                  if OPTIONS(1)==3, Hess, eig(Hess)', end
            elseif apx_g(j)==4
```

```
                    dgj = dgstore(:,j:ncon:j+(niter-1)*ncon);
                    for k=1:niter
                      dgy = dgj(:,k).*(-xstore(:,k).^2);
                      dg=[dg,dgy];
                    end
                    Hess = hapx( gstore(j,:)', dg, 1./xstore,alpha );
                    HessG(:,j) = Hess(:);
                    if OPTIONS(1)==3, Hess, eig(Hess)', end
              end
        end;
        if OPTIONS(1)==3, rank(xstore), end
%       HessF
%       Hess
%       disp('eig Hess')
%       eig(Hess)

        dx = constr('qapx', zeros(size(x0)), options, xlb-x, xub-x,
'qapxgrad',...
         f, g, gradf, gradg, HessF, HessG, x0, move, [apx_f apx_g] );
        % dx=sqp('qapx', zeros(size(x0)), options, xlb-x, xub-x, 'qapxgrad',...
        % [],[],f, g, gradf, gradg, HessF, HessG, x0, move, [apx_f apx_g] );
        x  = x0 + dx;

        [fapx,gapx] = qapx( dx, f, g, gradf, gradg, HessF, HessG, x0, move,...
         [apx_f apx_g] );
        countgrd = countgrd + 1;
        feaslp = max(gapx) <= options(4);
        if gapx(ncon+1) > active
              bound = ' bound';
        else
              bound = ' ';
        end

%     Evaluate functions at new point.
        [f, g(:)] = eval(fcnstr);
        [mg,mj]   = max(g);
        countfcn  = countfcn + 1;

        % Increase move limit if no constraints encountered and design improved.
        if feasible_start & f<fold & mg < 5*active
              if niter>1, move = move + movmax/movred; end;
        % Reduce move limit if linear approximation was feasible.
        else
              move = movmin + (move-movmin)/movred;
        end

%       Check convergence.
        if max(abs(dx))<OPTIONS(2) & abs(f-fold)<OPTIONS(3) & mg<OPTIONS(4)
              if OPTIONS(1)>0
                    disp([sprintf('%5.0f %12.6g %12.6g
%3.0f',countfcn,f,mg,mj)]);
                          disp('Optimization Terminated Successfully')
                          Active_Constraints = find(g>active)
              end
```

```
            OPTIONS(14) = niter;
            converged = 1;
      elseif niter >= OPTIONS(14)
            disp('Maximum number of iterations exceeded in MQA')
            disp('increase OPTIONS(14)')
            converged = 1;
      else
            apx_g = apx_order(2:ncon+1); % Restore approximation type after
first iter.
      end
end

% Return iteration history info in OPTIONS.
OPTIONS(10) = countfcn;
OPTIONS(11) = countgrd;
```

Appendix B: Routine For Calculating the Approximate Hessian

```
 function Hess = hapx( f, g, x, alpha, print )
%
% Quadratic Approximation: find approximation to Hessian
%                          based on current & previous information
%
% INPUTS
%
% f = function values column vector
% g = gradients (each column is a gradient of f)
% x = design points (each column is a point where f & g evaluated)
% alpha = restriction on negative terms 0 (no neg) to 1 (allows full neg)
% print = optional print control
%
% OUTPUT
%  Written by Robert A. Canfield
% Hess..... Hessian matrix approximation (n by n)
%
[ndv,npt] = size( x );
if npt < 2 | ndv < 1, Hess=zeros(length(x),length(x)); return; end;
%
% Local variables
%
% ndv=n.... Number of Design Variables
% npt=k.... Number of Points where f & g evaluated
% neq...... Number of Equality constraints
% N........ Number of independent terms in Hessian matrix
% P........ Set of design points where equality constraints met
% A........ LHS coefficient matrix of Hessian vector
% b........ RHS of equality constraints for function values
% x0....... Current (k_th) design point
% dx....... Change between previous and current design vectors
% diag1,n.. term in h where each upper diagonal of H begins/ends
% beta..... weighting factor for each previous design point
%
% Initialize local variables
%
if nargin<4
alpha =1;
end
alpha;
nprev = npt-1;
neq = nprev;
%neq = min( nprev, ndv );
N    = ndv*(ndv+1)/2;
P    = 1:nprev;
k    = npt;
x0   = x(:,k);
dx   = x(:,P) - diag(x0)*ones(ndv,neq);
```

```
b   = f(P) - f(k)*ones(neq,1) - dx'*g(:,k);
b  = max(b,alpha*b);
%
% Create index vectors to convert H from matrix to vector and back.
% (Upper diagonals of H are stored in a vector h).
%
ndiag = ndv;
diag1 = ones(1,ndiag);
diagn = zeros(1,ndiag);
diagn(1) = ndv;
for n=2:ndiag,
     diag1(n) = diagn(n-1) + 1;
     diagn(n) = diag1(n) + ndv-n;
end;
%
% Transformation matrix T converts n(n+1)/2 symmetric h vector
% to vector of rows of full H.
%
Trow = 1:ndv^2;
Tcol = zeros(1,ndv^2);
t = 0;
for row=1:ndv,
for col=1:ndv,
     t = t+1;
     u = abs(col-row);
     Tcol(t) = diag1(u+1)-1 + max(row,col)-u;
end;
end;
T = sparse(Trow,Tcol,1);
%
% Weighting factors for each design point.
%
beta = zeros(1,nprev);
for p=1:nprev, beta(p) = norm( x(:,p) - x0 )^2; end;
beta = max( beta ) ./ beta;
%
% Form A: linear coefficients of H in constraints, A h = b.
%         Each row of A corresponds to one of the P constraints.
%         Columns of A are coefficients of H in 1/2 dx'*H*dx
%         arranged by diagonal and then codiagonal terms.
%
A = zeros(neq,N);
for row=1:neq;
     dx = x(:,P(row)) - x0;
     A(row,1:ndv) = (dx.^2 / 2)';
     for n = 2:ndiag,
          col = diag1(n):diagn(n);
          I   = 1:1+ndv-n;
          J   = n:ndv;
          A(row,col) = dx(I)' .* dx(J)';
     end;
end;
%
% Error check against quadratic equations
```

```
%
for m=1:neq,
    p = P(m);
    dx = x(:,p) - x0;
    dg = g(:,k) - g(:,p);
    %
    % If slopes are same, function values must be consistent.
    %
%    slope0 = g(:,k)'*dx;
 %   slopep = g(:,p)'*dx;
%    if slope0*slopep >= 0 & abs(slope0*(f(p) - f(k))) <= eps,
 %       slope0, slopep, f(k), f(p)
%       error('Bad slopes')
 %   end
end;        % end for each previous point
%
%  Weighted equality constraints for function values and gradients.
%
   dx = ( x(:,P) - x0*ones(1,nprev) ) * diag(beta);
   dg = diag(beta)*(g(:,P) - g(:,k)*ones(1,nprev))';
%  Magnify equality constraints for function values vs. gradients.
   magnify = max(max(abs(dg))) / max([min(abs(b)), 10^(-8)]);
   magnify = max( magnify, 1 );
   AA = magnify * diag(beta)*A;
   bb = magnify * diag(beta)*b;
   %bb = [bb; dg(:)];
   for p=1:max(P);
   if dg(p,:)*dx(:,p)<0;
   dg(p,:)=alpha*dg(p,:);
   end
   end
   bb = [bb; dg(:)];
   for n=1:ndv,
       DelT = zeros(nprev,ndv^2);
       DelT(:,(n-1)*ndv+1:n*ndv) = dx';
       AA = [AA; DelT*T];
   end;
   h = pinv( AA ) * bb;
%
% Convert h to H.
%
Hess = diag( h(1:ndv) );
for n=1:ndv-1,
   codiag = h( diag1(n+1):diagn(n+1) );
   Hess = Hess + diag(codiag,n) + diag(codiag,-n);
end;
%
% Optionally check final residual and constraints.
%
if nargin<5, print=0; end;
if print>0,
   beta
   magnify
   wtdres = 0;
```

```
    constant = 0;
    for m=1:neq,
        p  = P(m);
        dx = x(:,p) - x0;
        dg = g(:,k) - g(:,p);
        R = Hess*dx + dg
        wtdres = wtdres + beta(p)*(R'*R)/2;
        constant = constant + beta(p)*(dg'*dg);
    end;
    A*h - b
    wtdres
    %residual = c'*h + h'*Q*h/2 + constant/2
end;
```

Appendix C: Routine for Calculating the approximate Functions

```
function [f, g] = qapx( dx, f0, g0, gradf, gradg, Hf, Hg, x0, move, apxtype )
%
% qapx      Quadratic Approximations to f and g
%
%-Inputs
%
% dx....... Change in independent variable vector
% f0....... Objective function value at base point dx=0
% g0....... Constraint vector at base point dx=0
% gradf.... Gradient of objective w.r.t. X
% gradg.... Gradients of constraints w.r.t. X
% Hf....... Hessian of objective w.r.t. X or Y (apxtype=2 or 4, respectively)
% Hg....... Hessian of constraints (each column is vectorized matrix)
% x0....... Original X vector
% move..... Spherical Move limit (not applied if move=0)
% apxtype.. Approximation Type (1=linear, 2=quadratic, 3=reciprocal,
4=rec.quad.)
%
%-Ouputs
%
% f........ Approximate function value
% g........ Approximate constraints
%            (and spherical move limit constraint in last entry if move>0)

%-Local Variables
%
% ndv...... Number of Design Variables
% ncon..... Number of Constraints
% dr....... Equivalent perturbation for 1st order Reciprocal approximation
% dy....... Change in intermediate (reciprocal) variable
% Hgj...... Hessian approximation for j_th constraint
% scale.... X relative scale factors for norm constraint
%
% Written by: Michael A. Blaylock and Robert A. Canfield
% Created:     8/12/94
% Modified:    2/ 4/95
%
% Modifications
% 2/4/95 - Handle nonpositive move limit.

dx = dx(:);
[ndv,ncon] = size( gradg );
if nargin<9, move=1; end;
if nargin<10
      apxtype=2*ones(1,ncon+1);
elseif any(apxtype>2)
      dr = dx.*x0./(x0+dx);
      dy = 1./(x0+dx) - 1./x0;
end
```

```
% Objective Approximation.
if apxtype(1)==1
      f = f0 + gradf(:)'*dx;
elseif apxtype(1)==2
      f = f0 + gradf(:)'*dx + dx'*Hf*dx/2;
elseif apxtype(1)==3
      f = f0 + gradf(:)'*dr;
elseif apxtype(1)==4
      f = f0 + gradf(:)'*dr + dy'*Hf*dy/2;
end

% Loop for each constraint to unpack Hessian.
for j=1:ncon
      if apxtype(j+1)==1
            g(j) = g0(j) + gradg(:,j)'*dx;
      elseif apxtype(j+1)==2
            Hgj = reshape( Hg(:,j), ndv, ndv );
            g(j) = g0(j) + gradg(:,j)'*dx + dx'*Hgj*dx/2;
      elseif apxtype(j+1)==3
            g(j) = g0(j) + gradg(:,j)'*dr;
      elseif apxtype(j+1)==4
            Hgj = reshape( Hg(:,j), ndv, ndv );
            g(j) = g0(j) + gradg(:,j)'*dr + dy'*Hgj*dy/2;
      end
end

% Add Euclidean norm move limit
if move>0
      scale = x0;
      small = sqrt(eps);
      for i=1:ndv
            if scale(i)<small,scale(i)=1; end;
      end;
      g(ncon+1) = ( norm(dx./scale) / move )^2 - 1;
end
```

Appendix D: Routine for Calculating the Approximate Gradients

```
function [df, dg] = qapxgrad( dx, f0, g0, gradf, gradg, Hf, Hg, x0, move,
apxtype )
%qapxgrad   Quadratic Approximations to f and g
%
% dx....... Change in independent variable vector
% f0....... not used
% g0....... not used (required for consistency with calling sequence of qapx)
% gradf.... Gradient of objective w.r.t. X
% gradg.... Gradients of constraints w.r.t. X
% Hf....... Hessian of objective w.r.t. X or Y (apxtype=2 or 4, respectively)
% Hg....... Hessian of constraints (each column is vectorized matrix)
% x0....... Original X vector
% move..... Spherical Move limit
% apxtype.. Approximation Type (1=linear, 2=quadratic, 3=reciprocal,
4=rec.quad.)

% Local Variables
%
% ndv...... Number of Design Variables
% ncon..... Number of Constraints
% mag...... Magnification for 1st order Reciprocal approximation gradients
% dy....... Change in intermediate (reciprocal) variable
% dydx..... Chain rule for reciprocal to direct variables
% Hgj...... Hessian approximation for j_th constraint
% scale.... X relative scale factors for norm constraint
%
% Written by: Michael A. Blaylock and Robert A. Canfield
% Created:     8/12/94
% Modified:    2/ 6/95
%
% Modifications
% 2/6/95 - Handle nonpositive move limit.

%--BEGIN
%
%  Initialize variables.
[ndv,ncon] = size( gradg );
dx = dx(:);
if nargin<9, move=1; end;
if nargin<10
      apxtype=2*ones(1,ncon+1);
elseif any(apxtype>2)
      mag  = (x0./(x0+dx)).^2;
      dy   = 1./(x0+dx) - 1./x0;
      dydx = -(x0+dx).^(-2);
end
if move>0
      dg = zeros(ndv,ncon+1);
else
      dg = zeros(ndv,ncon);
```

```
end

% Objective Approximation Gradient.
if apxtype(1)==1
      df = gradf(:);
elseif apxtype(1)==2
      df = gradf(:) + Hf*dx;
elseif apxtype(1)==3
      df = gradf(:).*mag;
elseif apxtype(1)==4
      df = gradf(:).*mag + (Hf*dy).*dydx;
end

% Loop for each constraint to unpack Hessian.
for j=1:ncon
      if apxtype(j+1)==1
            dg(:,j) = gradg(:,j);
      elseif apxtype(j+1)==2
            Hgj = reshape( Hg(:,j), ndv, ndv );
            dg(:,j) = gradg(:,j) + Hgj*dx;
      elseif apxtype(j+1)==3
            dg(:,j) = gradg(:,j).*mag;
      elseif apxtype(j+1)==4
            Hgj = reshape( Hg(:,j), ndv, ndv );
            dg(:,j) = gradg(:,j).*mag + (Hgj*dy).*dydx;
      end
end

% Euclidean norm move limit gradient.
if move>0
      scale = x0;
      small = sqrt(eps);
      for i=1:ndv
            if scale(i)<small,scale(i)=1; end;
      end;
      dg(:,ncon+1) = 2*(dx./scale.^2)/(move^2);
end
```

Bibliography

1. Barthelemy, Jean-Francois M. and Haftka, Raphael T. "Recent Advances in Approximation Concepts for Optimum Structural Design", NASA TM-104032, March 1991.

2. Canfield, Robert A. "A Rank Two Hessian Matrix Update For Sequential Quadratic Approximation," AIAA Paper 95-1334, 36th Structures, Structural Dynamics, and Materials Conference, New Orleans, LA, April 10-12,1995.

3. Canfield, Robert A. "Multipoint Quadratic Approximation for Numerical Optimization," AIAA Paper 94-4358-CP, 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Fl, September 1994, pp. 971-980, September 1994

4. Canfield, Robert A. "Design of Frames Against Buckling Using a Rayleigh Quotient Approximation," *AIAA Journal*, Vol. 31 No. 6, pp. 1143-1149, June 1993.

5. Canfield, Robert A. "High Quality Approximation of Eigenvalues in Structural Optimization," *AIAA Journal*, Vol. 28, No. 6, pp. 1116-1122, June 1990.

6. Canfield, R. A., Grandi R.V and Venkayya, V. B. "Optimum Design of Structures with Multiple Constraints", *AIAA Journal* Vol. 26, No. 1, pp. 78-85,1988.

7. Canfield, R.A., Grandhi, R.V., and Venkayya, V.B. Comparison of Optimization Algorithms for Large Structures, AFWAL-TM-86-204-FI:BR, Air Force Aeronautical Laboratories, WPAFB, Ohio, 5 May 1986.

8. Fadel, G.M., Riley, M.F., and Barthelemy, J.F.M., "Two Point Exponential Approximation Method for Structural Optimization," *Structural Optimization*, Vol. 2, pp. 117-124, 1990.

9. Fleury, Claude and Braibant, Vincent, "Structural Optimization: A new Dual Method Using Mixed Variable," *International Journal for Numerical Methods in Engineering*, Vol. 23, pp. 409-428, 1986.

10. Grace, Andrew, **Optimization Toolbox for Use With MATLAB**, Ch 3,pp. 9-11, 1993.

11. Haftka, Raphael T. and Gurdal, Zafer, **Elements of Structural Optimization**, Kluwer Academic Publishers, 1992.

12. Haftka, Raphael T., Nachlas, Joel A., Watson, Layne T., Rizzo, Thomas and Desai, Rajendra, "Two Point Constraint Approximation in Structural Optimization," *Computer Methods in Applied Mechanics and Engineering*, Vol. 60, pp. 289-301, 1987.

13. Kolonay, Raymond Michael, Structural Optimization of Large Scale Structures under Multiple Loading Conditions Subjected to Stress and Displacement Constraints, Masters Thesis, Ohio State University,1987.

14. Miura, Hirokazu and Schmit, Lucien A., Jr., "Second Order Approximation of Natural Frequency Contraints in Structural Synthesis," *International Journal for Numerical Methods in Engineering*, Vol. 13, No. 2, pp. 336-351, 1978.

15. Rassmussen, John, "Accumulated Approximation - a new method for structural optimization by iterative improvement," *Third Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, San Francisco, CA, September, 1990, pp. 253-258.

16. Schmit, Lucien A., Jr and Miura, Hirokazu. "Approximation Concepts for efficient Structural Synthesis," NASA CR-2552, March 1977.

17. Schmit, Lucien A. and Farshi, B "Some Approximation Concepts for Structural Synthesis," *AIAA Journal*, Vol. 17, No. 5, pp. 692-699, May 1974.

18. Snyman J. A. And Stander Nielen, "New Sucessive Approximation Method for Optimum Structural Design", *AIAA Journal*, Vol. 32, No. 1, June 1994.

19. Starnes, James H. and Haftka, Raphael T. "Preliminary Design of Composite Wings for Buckling, Strength, and Displacement Constraints", *Journal of Aircraft*, Vol. 16, No. 8, pp. 564-570, August 1979.

20. Toropov, V.V., Filatov, A.A., and Polynkin, A.A., "Multiparameter Structural Optimization Using FEM and Multi-Point Explicit Approximations", *Structural Optimization* 6, pp. 7-14, 1993.

21. Toropov, V.V. "Simulation Approach to Structural Optimization", *Structural Optimization* 1, pp. 37-46,1989.

22. Wang, Liping and Grandi, Ramana V. And Canfield, R. A., "Multipoint Hermite Approximation for Structural Optimization", AIAA Paper 94-4280-CP, 5th AIAA/ USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Fl, Sept. 7-9,1994, pp. 269-280.

23. Wang, Liping and Grandi, Ramana V., "Optimal Design of Frame Structures Using Multi-point Spline Approximation", *AIAA Journal*, Vol. 32, No. 10, pp. 2090-2098, October 1994.

24. Vanderplaats, Garret N. and Salajegheh, Eysa, "An Efficient Approximation Technique for Frequency Constraints in Frame Optimization," *Internal Journal for Numerical Methods in Engineering*, Vol. 26, pp. 1057-1069, 1988

25. Vanderplaats, Garret N., **Numerical Optimization Techniques for Engineering Design With Applications**, New York: McGraw-Hill, Inc., 1984.

VITA

Michael A. Blaylock was born September 28, 1962, in Chateauroux, France. He graduated from The Georgia Institute of Technology in 1984 with a bachelors degree in Civil Engineering. In June of 1984, he was commissioned a Second Lieutenant in the US Air Force and was assigned to Lackland Air Force Base, Texas. He later served a one year remote tour at Kunsan Air Base, Republic of Korea. After serving in Korea, he was assigned to Pope Air Force Base, North Carolina. While assigned to Pope AFB, he was deployed to the Persian Gulf during Operations Desert Shield and Desert Storm. He was awarded the Bronze Star for meritorious service during these campaigns. He was later selected to attend The Air Force Institute of Technology to pursue a masters degree in Astronautical Engineering.

Mike is married to the Andi D. Slone of Miamisburg, Ohio. They have a child due in May.